The Physics of High-Bulk Wool Batting

A thesis

submitted in partial fulfilment

of the requirements for the Degree of

Doctor of Philosophy

in Applied Physics

at

Lincoln University

by

J. A. H. Grigg

Lincoln University

2016

Abstract of a thesis submitted in partial fulfilment of the
requirements for the Degree of Ph.D.

The Physics of High-Bulk Wool Batting

by J. A. H. Grigg

Knoppy web is a product developed by FibreTech New Zealand Ltd, consisting of small spherical clusters of wool and PLA fibres suspended in a non-woven wool/PLA fibrous batt. It offers improved resilience against compression over regular wool batts, as well as better drape and washability. Knoppy web is targeted to compete with down clusters in bedding and apparel products.

As part of an ongoing commercialisation process, FibreTech New Zealand Ltd desired insight into the production process, optimised knoppy web specifications, and a physical understanding of knoppy web's resilience against compression. In this thesis we met these objectives, and tested several hypotheses about the benefits of both knops and PLA in non-woven fibrous batting.

We examined the process for producing knoppy web, extracted the relevant physics from each step, and linked theoretical insights to the available control strategies for various production parameters. We then examined the available literature on compression of spheres and random fibrous assemblies. Building upon these, we developed the first mathematical model describing the compression mechanics of a knoppy web.

In the model, a knoppy web is represented as a uniform array of knops embedded in a fibrous web. Each knop is treated as a hollow spherical membrane, to which a series of assumptions are applied such that the resulting sphere captures the core physical intuition of knop compression. The web is treated as a random fibrous assembly using van Wyk's equations. The model was developed using the energy method, and implemented using a variety of numerical techniques and computational packages.

A series of experiments were conducted to examine key parts of the knoppy web production process. It was observed that knops can provide significantly better compressional properties than both down clusters and goose feathers, and that they can be packaged at high strain for extended periods while retaining their desirable properties (after recovery via steaming). Additionally, an investigation of the parameter space for knoppy web specifications showed that there is significant tunability available in the properties of knoppy web.

The model was compared to compression curve data taken during the investigation of the specification parameter space. Although its predictive powers are limited, the model was able to provide satisfactory first-order fits to the experimental data.

Finally, the various theoretical and experimental results were used to develop optimum knoppy web specifications for use in overbody, underbody and apparel products. These specifications have been subsequently used by FibreTech New Zealand Ltd in the ongoing development of end-user products.

Keywords: knops; knoppy web; batt; fibrous assembly; fibre; wool; PLA; polylactic acid; bulk; compression; resilience; bedding; apparel; computer; simulation; model

To my parents.

For coping.

Acknowledgements

I would like to thank my supervisors, Prof Don Kulasiri, Dr Garth Carnaby and Prof Sandhya Samarasinghe, for all your support and guidance. This work was a veritable maze through a medley of academia and industry, and I would have been hard-pressed to find a more perfectly-suited group of guides.

I would also like to thank Mr Peter Sheldon and FibreTech New Zealand Ltd, both for the warm and accommodating industry environment, and for developing the commercial problem space in which this thesis resides. Special thanks go to Mr Harrie Stevens, Dr Dal Jung and Dr John McKinnon, from whom I have learned much about the inner workings of textile production.

I would like to acknowledge past and present members of the Centre for Advanced Computational Solutions research group, whom I have enjoyed working alongside. My thanks also go to the members of the staffs of Lincoln University and AgResearch Ltd who have supported my research, in particular Mr Ian Fowler and Dr Surinder Tandon.

Finally, I would like to thank Fiona, and my family and friends, for supporting me in my endeavours and being understanding in my absences.

Conflict of interest statement: FibreTech New Zealand Ltd was the nominated industry partner for this project, and some of the research was carried out on their premises; they had no influence over the final content of this thesis. The work reported within was funded in entirety by the Ministry for Business and Innovation (through a TechNZ Capability Grant), and the Wool Research Organisation of New Zealand (through Wool Industry Research Limited). Dr Carnaby has carried out research and consultancy work for FibreTech New Zealand Ltd over a number of years.

Contents

Abstract iii

Acknowledgements vii

Contents viii

List of Figures xiii

List of Tables xvii

List of Terms xix

1 Introduction 1

1.1 The wool fibre . . . 1
1.2 Knops and knoppy web . . . 3
1.3 Technical issues . . . 5
1.4 Hypotheses . . . 7
1.5 Thesis layout . . . 8

2 Physics of Processing 9

2.1 Introduction . . . 9
2.2 High level overview . . . 9
2.3 Carding . . . 11
2.3.1 Markov chains and the power of the doffer . . . 13
2.3.2 Withdrawal force for a fibre . . . 15
2.3.2.1 The effect of PLA on fibre migration . . . 16
2.3.3 Fibre breakage . . . 18
2.4 Cutting . . . 18
2.4.1 Transforming a fibre length distribution . . . 20
2.4.2 The effect of hooked fibres in a sliver . . . 24
2.5 Blending . . . 25
2.5.1 Carding . . . 25
2.5.2 Opening . . . 26
2.5.3 Pre-melling . . . 30
2.5.4 Cutting . . . 30
2.5.5 Number ratio of different fibre types . . . 32
2.6 Knopping . . . 32
2.7 Airlaying . . . 33
2.7.1 Target vs. actual weights . . . 36
2.8 Bonding . . . 36
2.9 Strategy and issues . . . 37

2.10 Summary . . . 39

3 Evaluation of literature 41
3.1 Introduction . . . 41
3.2 Micromechanical models of knops . . . 41
3.2.1 Hollow spherical spheres . . . 42
3.2.2 Solid spherical particles . . . 42
3.2.3 Discussion . . . 42
3.3 Micromechanical models of random fibre assemblies . . . 43
3.3.1 The van Wyk model . . . 44
3.3.1.1 Deformation of the bending element . . . 44
3.3.1.2 Counting the number of contact points . . . 47
3.3.1.3 Modelling the continuum strain . . . 48
3.3.1.4 The van Wyk equation . . . 48
3.3.1.5 Assumptions and issues . . . 49
3.3.2 Improvements on the van Wyk model . . . 49
3.3.2.1 The orientation density distribution function . . . 49
3.3.2.2 Randomly-oriented bending elements . . . 50
3.3.2.3 Slippage . . . 52
3.3.2.4 Steric hindrance . . . 53
3.3.2.5 Accuracy of the van Wyk constant . . . 55
3.3.3 Discussion . . . 55
3.3.4 van Wyk energy method . . . 57
3.4 Summary . . . 57

4 Knoppy Web Model 59
4.1 Introduction . . . 59
4.2 Simple sphere model . . . 59
4.2.1 Mapping between the uncompressed and compressed spheres . . . 61
4.2.2 Inner region . . . 63
4.2.3 Outer region . . . 67
4.2.3.1 Membrane energy from hoop strain . . . 69
4.2.3.2 Bending energy along the meridian . . . 70
4.2.4 Total energy . . . 71
4.2.5 Analysis . . . 72
4.2.5.1 Total energy . . . 72
4.2.5.2 Energy components . . . 73
4.2.5.3 Effect of parameters . . . 76
4.3 Knoppy web unit cell . . . 76
4.3.1 Basic unit cell . . . 80
4.3.1.1 Web volumes . . . 82
4.3.1.2 Total energy . . . 82
4.3.1.3 Analysis . . . 83
4.3.2 Modified unit cell . . . 93
4.3.2.1 Web volumes . . . 93
4.3.2.2 Total energy . . . 95
4.3.2.3 Analysis . . . 95
4.3.3 Discussion . . . 105
4.3.4 Simulating knoppy web . . . 105

4.4 Summary . . . 106

5 Experimental validation of knoppy web 109

5.1 Introduction . . . 109
5.2 Resilience after compression of various loose fills . . . 109
5.2.1 Method . . . 109
5.2.2 Results . . . 112
5.3 Long-term compression of knops . . . 112
5.3.1 Method . . . 113
5.3.1.1 Preparation . . . 113
5.3.1.2 Compression . . . 116
5.3.1.3 Recovery . . . 116
5.3.1.4 Force-displacement curves . . . 119
5.3.2 Results . . . 119
5.3.2.1 Compression . . . 119
5.3.2.2 Recovery . . . 121
5.3.2.3 Steaming . . . 121
5.3.2.4 Force-displacement curves . . . 125
5.3.3 Discussion . . . 125
5.4 Variations of the knoppy web blend . . . 129
5.4.1 Overview of Lots . . . 129
5.4.2 Methods . . . 131
5.4.2.1 Manufacture of Lots . . . 131
5.4.2.2 Bonding . . . 132
5.4.2.3 SEM . . . 132
5.4.2.4 Thermal resistance . . . 132
5.4.2.5 Stiffness . . . 133
5.4.2.6 Wash tests . . . 135
5.4.2.7 Force-displacement curves . . . 135
5.4.3 Results . . . 136
5.4.3.1 Qualitative assessment . . . 136
5.4.3.2 Indicative shrinkage during bonding . . . 136
5.4.3.3 SEM . . . 138
5.4.3.4 Thermal resistance . . . 144
5.4.3.5 Stiffness . . . 144
5.4.3.6 Wash tests . . . 146
5.4.3.7 Force-displacement curves . . . 149
5.4.4 Discussion . . . 151
5.5 Model comparison . . . 154
5.5.1 Modelling knoppy web parameters from production variables . . . 154
5.5.1.1 Knop size . . . 155
5.5.1.2 Knop stiffness . . . 156
5.5.1.3 Ratio of knops to web . . . 157
5.5.1.4 Ratio of wool to PLA . . . 159
5.5.2 Determining knop sizes . . . 159
5.5.2.1 Results . . . 161
5.5.3 Data preparation . . . 161
5.5.4 Knop model fitting . . . 161
5.5.4.1 Results . . . 161

5.5.5 Handling the fibrous behaviour of knops at high strain . . . 163
5.5.5.1 Results . . . 163
5.5.6 Knoppy web model fitting . . . 163
5.5.6.1 Fitting the van Wyk model . . . 166
5.5.6.2 Results . . . 166
5.6 Product development . . . 173
5.6.1 Development of specifications . . . 173
5.6.1.1 Overbody . . . 173
5.6.1.2 Underbody . . . 174
5.6.1.3 Apparel . . . 175
5.6.2 Optimum knoppy web specifications . . . 175
5.6.3 Methods . . . 176
5.6.3.1 Knoppy web production . . . 176
5.6.3.2 Sample products . . . 176
5.6.3.3 Stiffness . . . 176
5.6.3.4 Force-displacement curves . . . 181
5.6.3.5 Consumer trial of duvets . . . 181
5.6.4 Results . . . 181
5.6.4.1 Stiffness . . . 181
5.6.4.2 Force-displacement curves . . . 183
5.6.4.3 Consumer trial of duvets . . . 186
5.6.5 Discussion . . . 186

6 Conclusions and future work 189
6.1 Summary . . . 189
6.2 Validity of the hypotheses . . . 189
6.3 Avenues for future research . . . 190
6.3.1 Further development of the models . . . 190
6.3.2 Development of a knop model from a fibrous basis . . . 191
6.3.3 Model of a partially-bonded fibrous assembly . . . 192
6.3.4 The effect of wool blends on knops . . . 192
6.3.5 Theory of washing . . . 192

References 193

A The Lee Carnaby model 201
A.1 Overview . . . 201
A.2 Model description . . . 201
A.3 Analysis . . . 206
A.3.1 Verification of the re-implementation . . . 206
A.3.2 Issues . . . 212

B Code listings 219
B.1 `SimpleSphere.ipynb` . . . 219
B.2 `SimpleSphereParams.py` . . . 226
B.3 `BasicKnoppyWebUnitCell.ipynb` . . . 228
B.4 `BasicKnoppyWebUnitCellThickness.ipynb` . . . 237
B.5 `ModifiedKnoppyWebUnitCell.ipynb` . . . 246
B.6 `ModifiedKnoppyWebUnitCellThickness.ipynb` . . . 255
B.7 `SimpleSphereFit.ipynb` . . . 265

B.8 FibrousSimpleSphereFit.ipynb 269
B.9 ModifiedKnoppyWebUnitCellFit.ipynb 273
B.10 vanWykUnitCellFit.ipynb 281
B.11 Lee1992.ipynb 284
B.12 Supporting files 290
equations/ 290
basic_kw.py 290
factory_params.py 291
lee1992web.py 291
modified_kw.py 292
simple_sphere.py 292
vanwyk.py 294
plot.py 294
utils.py 298

List of Figures

1.1 The structure of a merino wool fibre. . . . 2
1.2 A pile of knops . . . 4
1.3 A sample of knoppy web . . . 4
1.4 Thermal bonding of bicomponent fibres. . . . 6

2.1 Outline of the general production process for knoppy web. . . . 10
2.2 Schematic diagram of a carding machine with two workers. Example teeth are shown to illustrate their relative orientations. . . . 14
2.3 Flow of fibres at a carding point. . . . 14
2.4 Markov chains with and without absorbing states . . . 14
2.5 States to be considered in the progress of the fibre through the card, with 1 swift and 4 carding points (I to IV). (The workers and the strippers are not shown in the figure) [30] . 17
2.6 Constructing the fibre withdrawal force from capstan forces. . . . 17
2.7 Schematic illustration of a fibre being removed from a tuft. The unshaded line indicates the shortest trajectory for the fibre's withdrawal when constrained by neighbouring fibres represented here by dots [34]. . . . 17
2.8 Uncarded fibre in contact with a carding element . . . 19
2.9 Geometry for calculating length fractions of cut fibres [47] . . . 22
2.10 Comparison of distributions for a sound crossbred wool of medium length (barbe = 100 mm) [47]. . . . 22
2.11 Probability densities resulting from the curring of slivers [44] . . . 23
2.12 Geometry for calculating length fractions of cut hooked fibres . . . 27
2.13 The average displacement of a fibre in a sliver as a function of the collecting power of the workers (p_w) and the doffer (p_d). ($\omega_i' = 10\,\mathrm{s}^{-1}$, $\omega_t = 200\,\mathrm{s}^{-1}$, $\omega_d = 10\,\mathrm{s}^{-1}$, $r_d = 0.4\,\mathrm{m}$.) . . . 27
2.14 The feedsheet of the opener. . . . 28
2.15 The spinner that distributes the opened fibres through the silo. . . . 28
2.16 Schematic showing how the feed conveyor belt on the opener can be used to blend fibres at small scales. . . . 29
2.17 Drawframe passages with eight-sliver feed and a draft of eight. Example of blending by doubling and drafting: drawframe blending [28]. . . . 31
2.18 A DOA airlay machine producing knoppy web. . . . 34
2.19 Close-up of the outlet of the DOA airlay. . . . 34
2.20 Schematic view of the DOA airlaying system [56]. . . . 35

3.1 A van Wyk pile of rods. . . . 46
3.2 The van Wyk bending element. . . . 46
3.3 Contact of fibre B of orientation (θ', ϕ') with A of (θ, ϕ) and sweepings of the former on both sides of the latter, keeping the contact point and direction of the former unchanged [58]. 51
3.4 Direction of fibre, and deformation of a fibre element [70]. . . . 51
3.5 Diagram of a slipping contact point [14]. . . . 54

3.6 The forbidden length. . . . 54
3.7 The evolution of strain energy within a van Wyk random fibre assembly. . . . 58

4.1 Schematics before and after compression for a knop of radius r_0 and shell thickness $2h$. . . . 62
4.2 General mapping of points between the uncompressed sphere and the compressed sphere. 62
4.3 Mapping a point P between the uncompressed sphere and the inner region of the compressed sphere. . . . 65
4.4 Mapping a point P between the uncompressed sphere and the outer edge region of the compressed sphere. . . . 68
4.5 Surface and contour plots of U_T. The lower blue line on the contour plot is $r' = c$, and the upper blue line is $r' = \frac{\pi}{2}c$. . . . 74
4.6 Strain energy of a sphere under compression. . . . 74
4.7 Surface and contour plots of U_{M_F}. The blue line is $r' = \frac{\pi}{2}c$. . . . 74
4.8 Surface and contour plots of U_{M_S}. The lower blue line on the contour plot is $r' = c$, and the upper blue line is $r' = r_0 \sin(\theta_c)$. . . . 75
4.9 Surface and contour plots of U_B. . . . 75
4.10 Energy components of a sphere under compression, plotted on (a) normal and (b) log-linear scales. Blue is $2U_{M_F}$, green is U_{M_S} and red is U_B. . . . 75
4.11 Minimised sphere energy for a range of r_0. . . . 77
4.12 Minimised sphere energy for a range of h. . . . 78
4.13 Minimised sphere energy for a range of μ_k. . . . 79
4.14 Compression of the basic knoppy web unit cell 81
4.15 Basic knoppy web model 85
4.16 Strain energy within the sphere, minimised along r'. . . . 85
4.17 Strain energy within the web, minimised along r'. . . . 86
4.18 Minimised basic unit cell energy for a range of K. . . . 86
4.19 Basic unit cell energy surfaces for a range of K, minimised along r'. . . . 87
4.20 Minimised basic unit cell energy for a range of ρ_w. . . . 88
4.21 Basic unit cell energy surfaces for a range of ρ_w, minimised along r'. . . . 90
4.22 Minimised basic unit cell energy for a range of t_0. . . . 91
4.23 Basic unit cell energy surfaces for a range of t_0, minimised along r'. . . . 92
4.24 Compression of the modified knoppy web unit cell 94
4.25 Modified knoppy web model 96
4.26 Strain energy within the sphere, minimised along r'. . . . 96
4.27 Strain energy within the web, minimised along r'. . . . 98
4.28 Minimised modified unit cell energy for a range of K. . . . 98
4.29 Modified unit cell energy surfaces for a range of K, minimised along r'. . . . 99
4.30 Minimised modified unit cell energy for a range of ρ_w. . . . 100
4.31 Modified unit cell energy surfaces for a range of ρ_w, minimised along r'. . . . 101
4.32 Minimised modified unit cell energy for a range of t_0. . . . 103
4.33 Modified unit cell energy surfaces for a range of t_0, minimised along r'. . . . 104
4.34 Strain gradients within the web component during compression 107

5.1 The various fills pre- (top) and post-compression (bottom). From left: down, goose feather, micro-knops, 27 µm wool knops, underbody knops. . . . 111
5.2 The hexagonal assembly of cans, and the components for the pistons. . . . 114
5.3 The fully-assembled can-and-piston compression aparatus. . . . 114
5.4 Loosely filling a can with knops. . . . 115
5.5 Can 5 being incrementally filled. . . . 115

5.6 Can 1 filled with 200 g of knops. . . . 117
5.7 The compression assembly just after the weights were applied. . . . 117
5.8 The compression assembly just before the weights were removed. . . . 118
5.9 Knops from cans 3 and 4 being steamed for 30 seconds. . . . 118
5.10 Apparatus and procedure for measuring the force-displacement curves of knops. . . . 120
5.11 The evolution of piston heights over the duration of compression. . . . 122
5.12 The evolution of piston heights as the compression weights were removed. . . . 122
5.13 Knops after removal of the weight and pistons. . . . 123
5.14 Side view of the unbonded knops from can 1 after removal of the weight and piston. . . . 123
5.15 Side view of the bonded knops from can 2 after removal of the weight and piston. . . . 124
5.16 Knops from cans 1 and 2 just over an hour after they had been emptied into bags. . . . 124
5.17 Force-displacement curves for the unsteamed knops. . . . 126
5.18 Force-displacement curves for the unbonded knops. The pre-compression data is from leftover uncompressed knops. . . . 127
5.19 Force-displacement curves for the bonded knops. The pre-compression data is from leftover uncompressed knops. . . . 128
5.20 Production process for the knoppy web lots. . . . 130
5.21 A "Shirley Stiffness Tester" designed for testing webs 134
5.22 Measuring with the Shirley stiffness tester 134
5.23 The compression apparatus for measuring force-displacement curves of knoppy web 137
5.24 Samples of Lot 6 (top) and Lot 2 (bottom), showing how the knops have been removed by the extra carding step. . . . 137
5.25 SEM image of Lot 2 web 139
5.26 SEM image of Lot 4 knop 140
5.27 SEM image of Lot 4 web 141
5.28 SEM image of Lot 1 web 142
5.29 SEM image of Lot 3 knop 143
5.30 Effect on the bending modulus q of (a) the knop:web ratio and (b) the PLA:wool ratio in the web. . . . 145
5.31 Wash test on untreated wool 147
5.32 Wash test on shrink-resist-treated wool 147
5.33 Wash test on polyester 147
5.34 Wash test on Lot 3 148
5.35 Wash test on Lot 5 148
5.36 Wash tests on other Lots 148
5.37 Effect on force-displacement curves of varying the knop:web ratio. . . . 150
5.38 Effect on force-displacement curves of varying the wool:PLA ratio. . . . 152
5.39 Effect on force-displacement curves of additional carding. . . . 153
5.40 The effect of web:knop ratio and t_0 on ρ_w. . . . 158
5.41 A 1 g sample of knops spread out on a surface for counting. . . . 160
5.42 A group of six knops. . . . 160
5.43 Fits of the knop model to data, shown on linear (left) and log (right) plots. Blue line is data, dotted line is fit with the initial values of the parameters, red line is best fit. . . . 162
5.44 Fits of the fibrous knop model to data, shown on linear (left) and log (right) plots. Blue line is data, dotted line is intial fit, red line is best fit. . . . 164
5.45 Components of the fibrous knop model fits. Dotted line is data, blue line is U_S, red line is U_W. . . . 165

5.46 Fits of the modified knoppy web model and van Wyk model to data for Lot 1, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit. 167
5.47 Fits of the modified knoppy web model and van Wyk model to data for Lot 2, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit. 168
5.48 Fits of the modified knoppy web model and van Wyk model to data for Lot 3, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit. 169
5.49 Fits of the modified knoppy web model and van Wyk model to data for Lot 4, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit. 170
5.50 Fits of the modified knoppy web model and van Wyk model to data for Lot 5, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit. 171
5.51 Fits of the modified knoppy web model and van Wyk model to data for Lot 6, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit. 172
5.52 Production process for overbody knoppy web. 177
5.53 Production process for underbody knoppy web. 178
5.54 Production process design for apparel knoppy web. 179
5.55 One of the baby mattresses produced from the underbody knoppy web. 180
5.56 The baby mattress contains three layers of underbody knoppy web sandwiched within a standard shell. 180
5.57 One of the single duvets produced from the overbody knoppy web. The duvet has been cut open to show the layers of knoppy web and fabric. 182
5.58 One of the quilted jackets produced from the apparel knoppy web. 182
5.59 Force-displacement curves for the unbonded overbody and underbody knops. 184
5.60 Force-displacement curves for overbody and underbody samples. 185

A.1 Configurations of a fibre segment before (OA_{1i}) and after (OA_{2i}) z-directional compression, and their projected chords OB_{1i} and OB_{2i} on the xy plane [71]. 202
A.2 Contour plot of θ_c. 204
A.3 Recreation of Lee Carnaby Figure 2. 207
A.4 Recreation of Lee Carnaby Figure 4. 208
A.5 Recreation of Lee Carnaby Figure 5. 209
A.6 Recreation of Lee Carnaby Figure 6. 210
A.7 Surface and contour plots corresponding to the Lee Carnaby figures. 211
A.8 Energy vs. Compressional strain 213
A.9 Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -0.1$. 214
A.10 Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -0.1$ and $\max(\nu_a) = 3$. 215
A.11 Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -1$ and $\max(\nu_a) = 3$. 216
A.12 Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -1$ and $\max(\nu_a) = 3$. 217
A.13 Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -1$ and $\max(\nu_a) = 3$. 218

List of Tables

2.1 Properties for some of the fibre types used in knoppy web production. 33
2.2 Number ratios for different fibre type blends. 33

4.1 Sphere model parameters 72
4.2 Parameters for the basic knoppy web model 83

5.1 Pre- and post-compression bulks. The micro-knop bulks are calculated with a sample weight of 57.4 g instead of 56.7 g. 112
5.2 Measured bulks of the knops prior to compression 113
5.3 Test compression of can 1. 116
5.4 Measured pre- and post-steam bulks of the unbonded (1, 3, 5) and bonded (2, 4, 6) knops. The percentage values for pre- and post-steam bulks are percentages of the measured bulks of leftover knops that had not been subjected to long-term compression (unbonded: $34\,\mathrm{cm^3\,g^{-1}}$, bonded: $38\,\mathrm{cm^3\,g^{-1}}$). The steam regain percentages are increases over the corresponding pre-steam bulks. 121
5.5 Overview of the various Lot variations, and the total wool fraction for each Lot. 131
5.6 Indicative shrinkage by area during bonding. 138
5.7 Inherent thermal resistance, warmth to mass and warmth to thickness ratios of Lots [95]. (μ = mean, σ = standard deviation, c_v = coefficient of variation) 145
5.8 Stiffness properties of the various Lots across the airlay 145
5.9 Comparison of stiffness properties 145
5.10 Wash test results. * = the Lots were wash tested after 22 months in storage. 149
5.11 Wash test comparisons 151
5.12 Calculated knop diameters 161
5.13 Initial values and ranges of the fitted parameters for the knop model 161
5.14 Fitted values for the parameters of the knop model 162
5.15 Initial values and ranges of the fitted parameters for the fibrous knop model 164
5.16 Fitted values for the parameters of the fibrous knop model 164
5.17 Values of the constants for the modified knoppy web model 167
5.18 Initial values and ranges of the fitted parameters for the modified knoppy web model . . . 167
5.19 Initial values and ranges of the fitted parameters for the van Wyk model 167
5.20 Fitted values for the parameters of the modified knoppy web model 168
5.21 Fitted values for the parameters of the van Wyk energy model 168
5.22 Optimum specifications for knoppy web blends. 175
5.23 Stiffness properties of the various Lots across the airlay 182
5.24 Weights and thicknesses used in calculation of figs. 5.60b and 5.60c. 183
5.25 Average rating change of the performance areas, and participant consensus [107]. The three consensus values represent the number of participants that rated the area higher (+), about the same (∼) or lower (-) after using the knoppy web duvet. 187

List of Terms

barbe The mean length of the fibres of a sliver, calculated from the proportions by mass of the fibres in the sliver.

batt A three-dimensional non-woven assembly of fibres used for lining or insulating items.

hauteur The mean length of the fibres of a sliver, calculated from the proportions by linear density of the fibres in the sliver.

knop A small spherical cluster of fibres.

knoppy web A batt containing some percentage of knops.

linear density The mean mass per unit length of fibre or sliver.

sliver A long, untwisted ropelike bundle of fibres with a roughly uniform thickness.

Chapter 1

Introduction

1.1 The wool fibre

Wool is a textile fibre with a history of use going back to antiquity [1]. It is naturally-grown, easily harvested, and can be produced sustainably. Once harvested it can be processed into a variety of forms, most notably into woollen and worsted yarns that are subsequently used in the production of fabrics and clothing. It is very much a niche fibre in today's global textile industry, making up about 1.3% of total fibre usage in 2014—but this still amounts to around 1.2 million tons [2]. Aside from the above factors (and its abundance both within New Zealand and overseas), its continuing popularity can be attributed to the physical characteristics of the fibre, which give rise to several useful properties:

- *Felting:* Wool fibres have a two-component structure consisting of a central cortex with scaly cuticlar cells surrounding it (fig. 1.1). The scaly structure increases the friction between adjacent wool fibres, and creates a ratcheting effect where fibres will migrate in one direction through an assembly. This gives wool its felting ability, where the fibres become irreversibly entangled and form mats of fabric.
- *Insulation:* The core itself has a further bilateral aspect responsible for its naturally-occurring crimp [3, 4]. The presence of the crimp creates space between adjacent fibres, trapping air within the overall structure. This makes wool particularly efficient at insulating.
- *Hygroscopicity:* Wool fibres can absorb up to 30% of their weight in water without feeling damp [5]. Wool bedding products will therefore absorb moisture as the humidity of the surrounding air rises (such as from perspiration during sleep), and are perceived to be significantly drier by users than cotton or polyester [6].
- *Elasticity:* Wool fibres have remarkable elasticity when wet—individual fibres can stretch up to 30% and then return to their original length [7]. Dry wool is viscoelastic (length recovery is incomplete), and at room temperature shows stress-relaxation and creep [8]; however this can be reversed by immersion in water [7], or raising the temperature above the glass transition temperature [9].
- *Fire-resistance:* Compared to other commonly-used textile fibres, wool is the most flame-resistant because it has naturally high nitrogen and water content. It therefore has a very high ignition temperature and requires high levels of oxygen to sustain combustion [10]. Wool is also self-extinguishing, because it forms an insulating surface char as it burns that protects it from further oxidation [11, 12].

These properties make wool an ideal fibre for everything from suits and hats to jerseys and carpets, depending on the particular fibre diameter. However, these properties also pose several challenges for some types of products. In particular, the high felting ability and bending stiffness create significant issues when attempting to use wool in batting or other non-woven products.

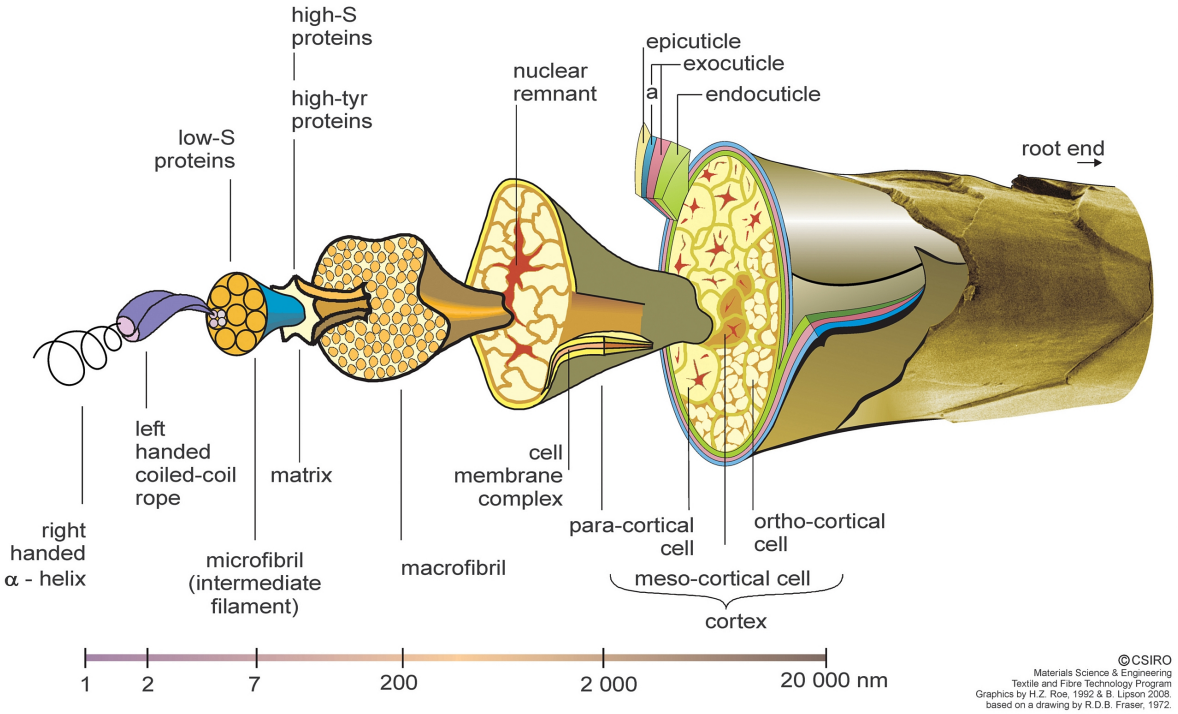


Figure 1.1: The structure of a merino wool fibre.

A wool batt or non-woven textile can be likened to a "pile of logs," where the majority of fibres are aligned perpendicular to the normal of the batt. When compression is applied perpendicular to the plane of the batt, energy is reversibly stored as bending strain in the wool fibres—but there is also significant fibre-fibre slippage [13, 14]. This slippage is essentially irreversible, due to fibre locking by friction (between fibres) and the aforementioned stress-relaxation of wool fibres [15]. Over time, this leads to a considerable loss of bulk and compaction of the product. A common example is a wool underlay on a bed, which becomes flatter and more uncomfortable the longer it is used. The physics underlying this situation has been extensively studied previously (see e.g. the mechanics of flattening of carpets [16]).

Meanwhile, the bending strain stored within the wool fibres enables the batt to support its own weight. This prevents the batt from effectively draping. Using the above example of a wool underlay, in addition to becoming hardened with use, the underlay tends to stick out the sides of the bed instead of hanging down.

Finally, wool products have a reputation for being impossible to wash safely in a washing machine. This is another side-effect of the high felting ability of wool. The repeated harsh working of wool products in a washing machine has a similar effect to the uniaxial compression considered above, but omnidirectionally. The effect is that the wool products felt and shrink under washing. Chemical treatments are available that can provide a "shrink-resistant" coating for the wool [17–19]. For non-woven batts, however, this makes the situation worse—the washing machine destroys the integrity of the product (see fig. 5.32 in section 5.4.3.6 for an example).

Down-filled products set the benchmark for performance in underbody and overbody uses. Down clusters are effective at insulating [20], have exceptional loft [20, 21], and can be washed (gently) without drastically affecting the integrity of the product [22]. Any manufacturer of wool-filled product who wants to compete in these sectors needs to be able to replicate the performance of down-filled products.

1.2 Knops and knoppy web

In the early 1980s, an alternative method and technology for processing wool fibres was developed at the Wool Research Organization of New Zealand (WRONZ). The resulting product consists of small clusters of entangled wool fibres called "knops," and they present several key advantages over traditional fibrous masses:

- As outlined in the previous section, a fibrous mass is prone to flattening after compression; repeated compression cycling results in bulk loss. Knops have been qualitatively observed to be very resilient to compression; it is hypothesised that the reason for this is that knops are more elastic and experience a higher ratio of stored elastic energy to work against friction. Knops have a much lower percentage of fibres aligned perpendicular to the direction of compression than a fibrous mass; therefore, more of the compressive strain is translated into bending strain of the constituent fibres.

- Knops are small clusters of fibre, usually less than 1cm in diameter, and are therefore not affected by the long-range bending strength of a wool batt—a handful of knops has no concept of drape. A transversely isotropic product filled with these knops (or, for instance, a duvet) would therefore have little to no resistance to bending.

FibreTech New Zealand Ltd is a New Zealand company based in Woolston, Christchurch. It was established in 1985 by Mr. Peter Sheldon, specifically to take the inital knop technology from WRONZ to the point where it could be leveraged commercially. Over the last several decades it has developed and refined the knop manufacturing process, and it is now possible to create knops to a wide variety of specifications (e.g. size, softness, presence of a "tail").

FibreTech New Zealand Ltd's answer to the problematic wool batt is a product called "knoppy web," which consists of knops embedded in a wool batt (the "web"). FibreTech New Zealand Ltd uses a knop:web

Figure 1.2: A pile of knops

Figure 1.3: A sample of knoppy web

ratio of around 80:20 by weight, so for an equivalent total product weight there is much less web fibre than a regular wool batt. The overall mechanical properties of knoppy web have been qualitatively observed by FibreTech New Zealand Ltd to be generally superior to a plain wool batt, specifically in regards to resilience under compression.

Thermal bonding of non-woven fabrics has been widely used in industry as a means of adding cohesion and stability to products for several decades [23]. This is a process where a binding fibre, powder or web is blended into the fibre supply of the product, and then the end product is heated up above the melting point of the surface of the bonding fibre. The binder polymer subsequently melts and sticks the surrounding fibres together. For wool-based products (such as insulation batts) a bicomponent polyester fibre is generally used for melt bonding [24]. This creates a scaffold of polyester bonded to polyester fibre—the wool itself does not bond to the polyester.

FibreTech New Zealand Ltd has used thermal bonding to innovate further upon the base idea of a knoppy web: they blend a bicomponent polylactic acid (PLA) fibre into the wool supply. Unlike other oil-based synthetic binder polymers, PLA is a biodegradable [25] thermoplastic derived from renewable resources [26], and is completely recyclable [27]. The bicomponent PLA fibre that FibreTech New Zealand Ltd uses has a central PLA core with a melting temperature of around 160°C, surrounded by a PLA sheath with a melting temperature of around 130°C (fig. 1.4a). Thus any PLA fibres that are in contact in the product (fig. 1.4b) can be thermally bonded together when heated up to above 130°C. At this temperature, the outer sheath of the PLA fibres melts but the integrity of the inner core remains (fig. 1.4c); this allows PLA-PLA bonds to form between adjacent fibres. Once cooled, the fibres are joined with a rigid bond (fig. 1.4d), creating a three-dimensional substructure within both the knoppy web and the knops themselves.

FibreTech New Zealand Ltd is able to manufacture knops containing a percentage of PLA fibre, and by blending PLA into the web they can create a wool/PLA knoppy web product. FibreTech New Zealand Ltd uses a wool:PLA ratio of around 85:15 by weight for most products, but because the PLA fibres are generally finer, the number ratio is significantly closer to equality (this is discussed in more depth in section 2.5.5). When sufficiently blended, this means that a significant fraction of wool fibres will be separated from neighbouring wool fibres by PLA fibres.

In-house qualitative testing has indicated that the presence of PLA improves the performance of the end product, both under compression and in washing. It is hypothesised that the reason for this is multi-faceted:

- *The PLA improves resilience against compression.* The finer PLA fibres build up less internal energy when bent. But the three-dimensional bonded substructure within the knoppy web inhibits relaxation of this energy through fibre slippage, which would occur more readily in the case of unbonded fibres.

- *The PLA forces separation of the wool fibres and restricts fibre migration, reducing its felting ability.* This is examined in section 2.3.2, and a mechanism is proposed in section 2.3.2.1.

1.3 Technical issues

The knoppy web products that FibreTech New Zealand Ltd have developed are in the process of being perfected. Prior to the start of this thesis, there were several open issues that required resolution in order for an effective commercial product to be developed and marketed.

Firstly, the blending process needed improvement. The end knoppy web product is sometimes observed to be "patchy" in places, and some regions will contain more knops while others will contain more web. This is partly a result of the age of FibreTech New Zealand Ltd's machinery, but the initial process used to create the knoppy web blend also contributes to the imperfections. The opener and card at the factory

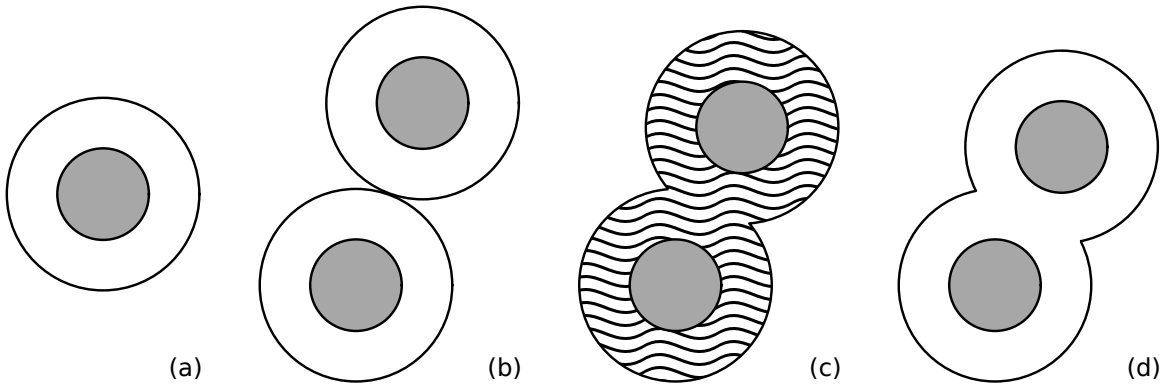


Figure 1.4: Thermal bonding of bicomponent fibres.

are also not optimised for processing the finer PLA fibre. An understanding was required of how the blending efficiency affects the performance of the knoppy web.

Secondly, the knoppy web product needed to be optimised for different uses. As mentioned in the previous section, FibreTech New Zealand Ltd has developed the ability to customise knops to a variety of specifications. Similar work needed to be done for the knoppy web product, as different uses have different requirements. The three key categories for knoppy web products are underbody, overbody and apparel. Underbody products need to withstand the repeated cycling of applied weight during normal usage (such as people sleeping); compression resistance and bulk retention are perceived to be the important properties. For overbody usage, bulk retention of the knoppy web under its own weight is key for warmth, and drape is important for comfort over the body. Bulk retention matters in apparel as well (again, for warmth), but much more important in the minds of consumers is softness of feel; no one wants a lumpy jacket.

Finally, an understanding of how the knoppy web works on a fundamental level was required. All prior understanding within the company had been gleaned from qualitative observations of trials. Each trial has a not insignificant cost, and FibreTech New Zealand Ltd only has a single production line. Hence, a piecewise iterative approach has underpinned the development of the knoppy web product thus far. Such an approach has been adequate for the early development stage, where there are too many variables to even quantify, let alone study. The knowledge/IP that the company has built up around the "feel" and handle of the knoppy web product has been important to current progress. However at this point in the product development process a more quantifiable approach to the creation and analysis of the knoppy web product was required, to better determine the key factors in both manufacturing different styles of product, and reliably manufacturing the same style of product. At the same time, a theoretical analysis of knoppy web would provide greater understanding of the various mechanisms operating within the knoppy web—from the effect of the knops on the overall product, to the bonded PLA substructure, to the effect that the PLA fibres have on the separation of the wool fibres and the felting ability. Such understanding is of particular importance when considering the requirement for usage optimisation outlined above. In addition, there are some markets where a solid scientific grounding for a product is highly beneficial for marketing it to the end consumer, Japan being the notable example.

1.4 Hypotheses

The problem space that this PhD resides within was outlined in the previous section. From this problem space, and the existing work that had been done by FibreTech New Zealand Ltd, several hypotheses could be posed with regard to the benefits that the wool/PLA knoppy web product might provide over the industry standard wool batt:

Hypothesis 1. *A knoppy web has higher bulk retention than an equivalent weight of plain wool batt, due to less fibre slippage and more compression energy being stored in the knops.*

Hypothesis 2. *A knoppy web has better drape than an equivalent weight of plain wool batt, due to the lower quantity of fibres aligned within the plane of the batt.*

Hypothesis 3. *A product containing a non-zero percentage of bonded PLA fibre has higher bulk retention than a product containing zero PLA fibre, due to the stiffness created by the bonded PLA substructure and the restriction on slippage of non-bonded fibres.*

Hypothesis 4. *A product containing a non-zero percentage of bonded PLA fibre has better performance and resistance to commercial washing machines than a product containing zero PLA fibre, due both to the shrink-resistance of the bonded PLA substructure, and to the lowered felting ability of the wool fibres caused by their forced separation with the blended PLA fibres.*

These hypotheses form the base upon which this PhD is built. Validating (or invalidating) these hypotheses will provide a major contribution to the ongoing commercial development process that FibreTech New Zealand Ltd is undertaking, and by extension will benefit the NZ wool industry inasmuch as the behaviour of these wool/PLA knop-based products will be better understood.

1.5 Thesis layout

The remainder of the thesis is presented as follows:

Chapter 2 examines the process of creating knoppy web as a whole, elucidates the physics which underlies the various processing steps, and explains the origins of the various properties.

Chapter 3 reviews the background literature necessary to understand the behaviour of the knoppy web, focusing on the physics of compression of spheres and random fibrous assemblies.

Chapter 4 presents the first proposed model of the compression mechanics of a knoppy web. The model is based on a simplified model of the compression mechanics of an individual knop (motivated by observations made in chapter 3).

Chapter 5 presents the experimental work conducted throughout the product development process, fits the knoppy web model to experimental compression curves, and develops optimum knoppy web specifications for several product categories.

Chapter 6 concludes with a summary of the work presented in this thesis, and suggestions for future avenues of research and development.

Chapter 2

Physics of Processing

2.1 Introduction

Chapter 1 outlined the practical need for understanding the properties of knoppy web. An important part of this is understanding the process by which the knoppy web is created, and the variables that are available for tuning its properties. Knoppy web is envisaged as a versatile product, but that versatility is literally baked into the knoppy web at production time. Improvements in the factory's ability to reliably control product properties will directly translate into commercial success.

In this chapter, we outline the production process that has been developed for creating knoppy web. Each processing stage is then examined in turn, and the relevant physics is extracted and discussed. The theoretical insights are then linked back to the available control strategies.

Key original contributions in this chapter:

- A proposed mechanism for reduced fibre migration in knoppy web (section 2.3.2.1).
- A previously unstated measure of the macroscale blending power of carding (section 2.5.1).
- An analysis of the number ratio of different fibre types (section 2.5.5).

2.2 High level overview

Over the years that knoppy web has been developed, various processes have been identified for creating different products. Most of the intellectual property is in the recipe used; while there are some operational differences, the majority of the process is well-established and generalised.

All knoppy web recipes require wool and PLA. Most recipes require at least two types of wool. Some recipes also use different types of PLA for the knops and web. There are three key quantities in the recipe:

1. The ratio of wool to PLA by weight in the knops.
2. The ratio of wool to PLA by weight in the web.
3. The ratio of knops to web by weight in the knoppy web.

Some recipes use different wool:PLA ratios in the knops and web. This usually coincides with a bulk stock of knops being produced, that is then used later in the production of several different knoppy web products. Other recipes use the same wool:PLA ratio for the knops and web; this is better for smaller production runs, because the fibre blend can be created once and then split between knopping and web fibre.

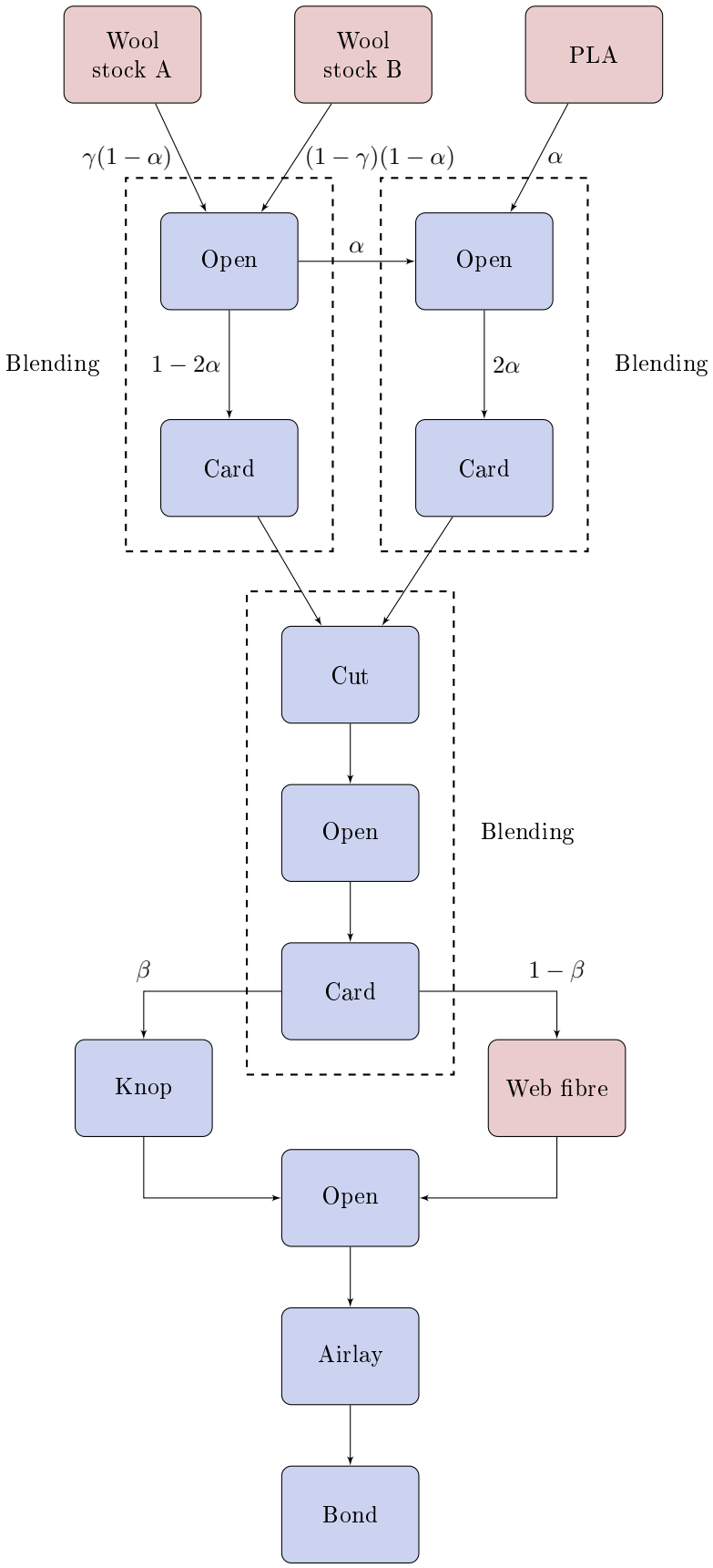


Figure 2.1: Outline of the general production process for knoppy web.

Figure 2.1 presents an example outline of a process for producing knoppy web using two wool components and a PLA component, with the same wool/PLA blend in the knops and web. α is the fraction of PLA by weight in the knoppy web, β is the fraction of knops by weight in the knoppy web, and γ is the ratios by weight of the two wool components. The process has the following steps:

1. The various wool components are weighed out and opened together to form the base wool stock.
2. The PLA fraction α is weighed out, and opened together with an equal weight of wool stock.
3. The 50:50 PLA:wool blend is then carded as one single lot, as is the remaining wool stock.
4. The two carded slivers are cut together.
5. The cut sliver blend is opened and carded to form a final fibre blend.
6. A fraction $1 - \beta$ of the fibre blend is weighed out and put aside to form the web component.
7. The remaining β fibre blend is knopped.
8. The knops are opened together with the set-aside fibre to form the knoppy web blend.
9. The blend is sent to the airlay machine to form the knoppy web product.
10. The knoppy web is bonded.

There are several distinct elements to the production process: carding, cutting, blending, knopping, airlaying and bonding. Each of these steps will now be considered in turn.

2.3 Carding

Carding is a fundamental step in the processing of raw wool into usable products. Wool naturally clumps together during its growth into staples, trapping dirt and resulting in a non-uniform distribution of fibres. In order for wool to be a useful product component, it must be opened up to allow the fibres to mix and dirt to fall out, then condensed into a form from which it can be efficiently processed. Carding machines are the technological legacy of centuries of refining this process.

Figure 2.2 is a representative diagram of a carding machine, showing the various operating elements. The basic principle behind it (along with several similar wool processing devices [28]) is one of expansion and contraction:

- Fibre staple is fed into a slow-moving pair of rollers (the nippers, N), creating a controlled stream.
- Fibres from the nippers are picked up by a fast-moving drum (the swift), covered in small teeth pointing in the direction of rotation. The difference in speed between the nippers and the swift drastically decreases the density of the stream, forming a thin film of fibres on the surface of the swift.
- The fibre is then collected by a slow-moving drum (the doffer), which transports it to the exit of the carding machine where it is collected as a long bundle (a sliver).
- Between the nippers and the doffer are a series of smaller slow-moving drums (the workers) that lift some of the fibre off the surface of the swift, and deposit it back at an earlier point (via the strippers).

The teeth on the workers and doffer point away from the direction of rotation, so that at the point where the swift and worker/doffer meet (the carding point), the teeth pull on fibres in opposite directions. This provides the force necessary to extract fibres out of tufts, and also enables the swift to "comb" the fibres on the workers and doffer to provide a degree of straightening.

A crucial aspect of carding is its ability to separate and mix fibres. This is important both to ensure that the resulting sliver is homogeneous, and to aid with blending fibres from various sources; bulk wool fibre can contain non-trivial variations in properties caused by both the growth environment and earlier processing stages. The separating and mixing comes from the action of the workers and doffer. The workers remove fibres from the swift and deposit them at an earlier point, which provides fibre mixing within a nearby region of the web; meanwhile the doffer collects only some of the fibres from the swift, leaving the remainder to cycle past the workers again and mix with additional incoming fibres from the feed sheet.

The effect of the workers and doffer on the carding of wool was examined in a seminal paper by Martindale [29]. Figure 2.3 shows the paths that fibres can take around a carding point. Martindale defined the "collecting power" of a worker as the fraction of fibre reaching a carding point (the point of contact between the worker and swift) that is picked up by the worker, and also pointed out that the collecting power is equivalent to the probability that a fibre approaching the worker would be picked up. The collecting power of the doffer is defined similarly. Collecting power increases noticeably as worker speed increases, and is also affected by the position of the worker and the closeness of the worker to the swift [29]. The increase in collecting power with worker speed is attributed to the fact that more "clean" worker teeth are presented to the swift in a given interval, decreasing the chance of the worker teeth becoming clogged [29]. However, the increased removal rate of fibres from the carding point also decreases their exposure to the combing effect of the swift, lowering the chance of lifted fibres being re-captured.

The collecting power of a worker can be roughly determined by measuring the amount of fibre m_i on the worker (after stopping the card mid-production[1]). The collecting power of the worker is then

$$p_i = \left[\frac{\nu \phi_i}{m_i \omega_i} + 1 \right]^{-1} \tag{2.1}$$

where ν is the production feed rate, ϕ_i is the angle over which the collected fibre was spread on the worker, and ω_i is the speed of the worker. This method is simplistic in that it ignores the fibres that escape the doffer and cycle round again to the worker; thus it becomes less accurate as the collecting power of the doffer decreases and more fibre is left on the swift.

One of the factors that influences a choice of fibre blend is the particular cards available. The teeth on the card are obviously not easily altered, and the size of the teeth is generally considered an invariant per card. There are therefore limits on the diameters of fibre that a card can effectively process:

- If the fibres are too fine, tufts will not be effectively split apart, and the product will contain clumps. The card will also lose more fibres out the bottom.
- If the fibres are too coarse, the teeth will be less effective at collecting the fibres.

Most factories dealing with multiple fibre diameters will have several cards to use at the appropriate times. However, when dealing with fibre blends containing multiple fibre diameters that are significantly different (as can be the case for some knoppy web blends), a compromise must usually be decided on and accounted for; processing a blend on a different card could produce different results.

[1] The collected fibre will of course not have been entirely deposited on the worker at actual speed, but the discrepancy is small enough [29].

2.3.1 Markov chains and the power of the doffer

Carding is an inherently random process—it is intractable to predict the exact form of the fibres within a carded sliver, or their exact trajectory. Fortunately, stochastic mathematics (in particular, stochastic process theory) can be used to elucidate useful generalised information about the carding process. One branch of stochastic mathematics in particular is directly applicable to carding, due to the nature of the machinery: Markov process theory.

A Markov process is defined as a "memoryless stochastic process." "Stochastic" means that the future state of the process cannot be explicitly determined, but instead can be predicted based on a set of well-defined probabilities; "memoryless" implies that these probabilities only depend on the current state, and not on how the system arrived at this state. Thus Markov processes are a special class of processes whose behaviour can be completely and precisely specified given any initial or intermediate state.

A Markov chain simplifies the picture further, by restricting the system to a finite number of well-defined states. Figure 2.4(a) shows an example of a system with four states, and a set of possible transitions from one state to another. A Markov chain describes the probabilities of the system transitioning between these different states.

If a particular state has no outbound transitions (fig. 2.4(b)), it is said to be an absorbing Markov state. For a system that contains absorbing Markov states, the probability that the system will transition into one of these states approaches 1 as time progresses.

Carding can be effectively modelled as a Markovian process. Monfort [30] showed that the card can be represented as a Markov chain, by considering that the carding points act as transitions between states. If we assume that a fibre arriving at a carding point will be lifted or not at random, then the path the fibre takes (being lifted by the worker or staying on the swift) clearly depends only on that fibre arriving at the worker[2]. This is exactly the condition required of a Markovian process.

Figure 2.5 shows a schematic for a carding machine with four workers. similar to fig. 2.2, but now labelled with various states E1 to E7. A fibre will enter the swift at point T, after which it progresses around through the various carding points I to IV. At each point, the collecting power of the worker determines whether the fibre is lifted by the worker or continues on. If the worker does lift the fibre, it will deposit it ahead of its carding point. Therefore, each carding point can be represented by two transitions:

- A transition from the current state to the next state.
- A transition from the current state to itself.

For example, a fibre in state E1 will reach carding point I and transition either to E1 (with probability equal to the carding power p_1) or to E2 (with probability $1 - p_1$).

When a fibre in state E5 reaches the doffer (point P), it can transition either to E6 (on the doffer) or E7. This is different to the carding points, because the doffer never returns the fibre to the swift. Thus, the doffer state E6 can appropriately be labelled as an absorbing Markov state. The state E7 returns the fibre to carding point I (the point T never removes fibre, so there is no interaction); therefore it is equivalent to state E1.

The primary metric for the power of a card is the amount of time that fibres spend inside it; fibres will undergo more carding and blending the longer that they stay in the card. For the Markov chain shown in fig. 2.5, the mean total number of states that a fibre will transition through before exiting the card at point P is [30]

$$t = \frac{1}{p_d}\left(1 + \sum_{1}^{4} \frac{1}{q_i}\right), \tag{2.2}$$

[2]In reality, there will be some deviation from a Markovian process, e.g. when a fibre is still caught in an unopened cluster. However, on average, a Markovian process is suitable.

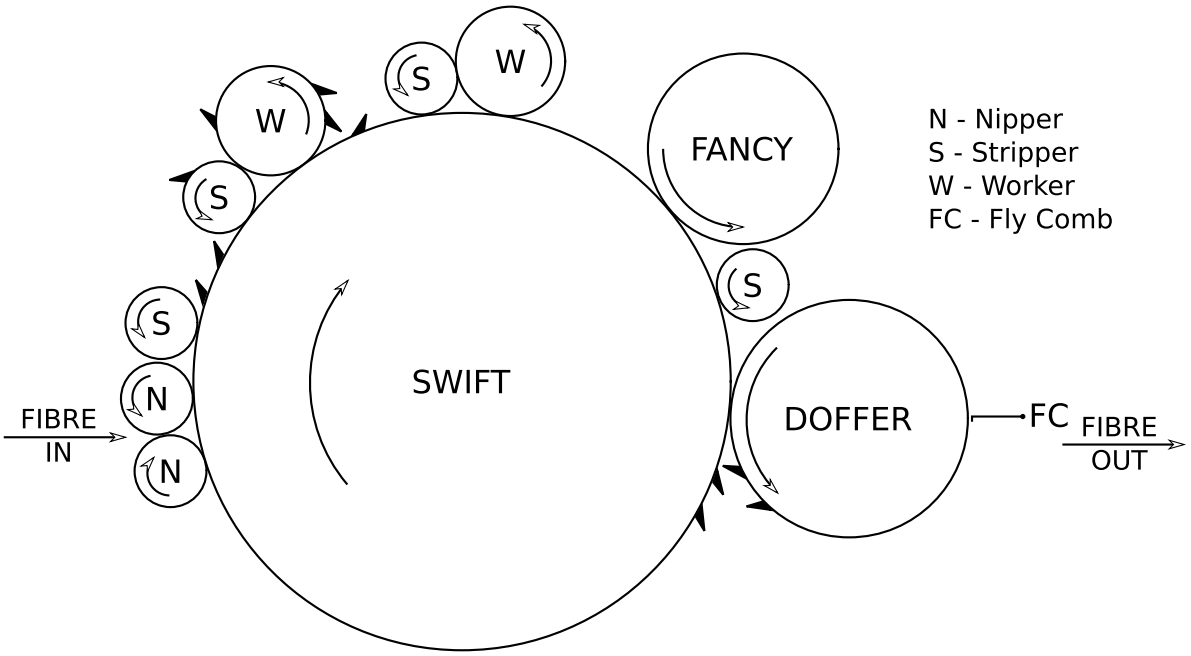


Figure 2.2: Schematic diagram of a carding machine with two workers. Example teeth are shown to illustrate their relative orientations.

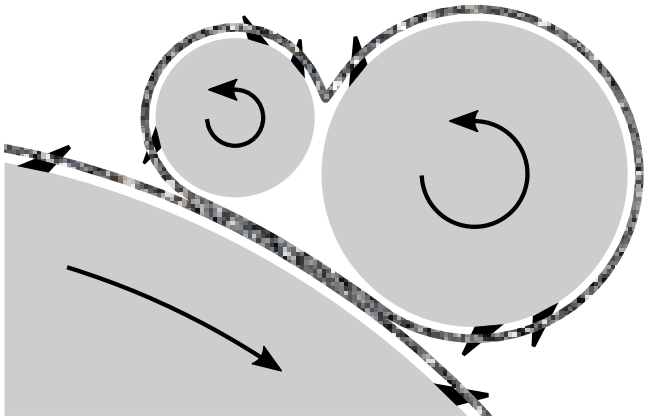
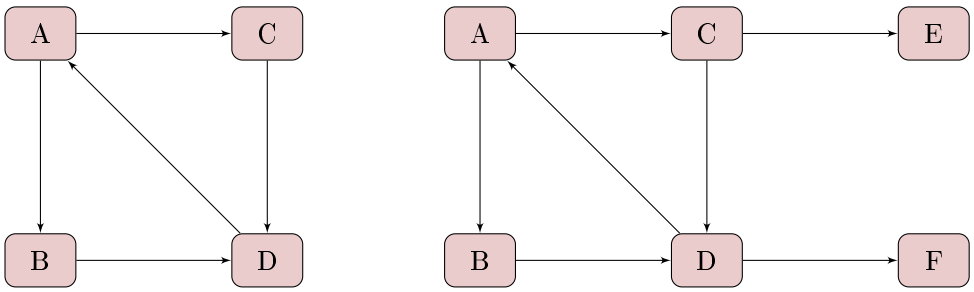


Figure 2.3: Flow of fibres at a carding point.



(a) A Markov chain

(b) A Markov chain with absorbing states

Figure 2.4

where $p_i = 1 - q_i$ is the collecting power of carding point i, and p_d is the collecting power of the doffer. From this, it can be determined that the mean time a fibre spends inside the card (from entering at point T to exiting at point P) is

$$\theta = \frac{1}{p_d}\left((p_d\alpha + q_d)\frac{1}{\omega_t} + \sum_1^4 \frac{p_i}{q_i\omega_i'}\right), \tag{2.3}$$

where $q_d = 1 - p_d$, α is the fraction of a turn of the swift corresponding to the arc from point T to point P, ω_t is the angular velocity of the swift, and $t_i = 1/\omega_i'$ is the time it takes for a fibre to travel from carding point i around the worker, stripper and swift before returning to carding point i.

It is clear that that the blending power of a card is heavily influenced by the doffer—the mean time a fibre spends inside the card is proportional to $\frac{1}{p_d}$. Increasing the collecting power of the workers does increase ω, but decreasing p_d gives the workers more "bites at the cherry."

2.3.2 Withdrawal force for a fibre

The previous section showed how carding can be treated at an aggregate stochastic level, and how the behaviour of the card depends on the collecting power of the workers and the doffer. On a micromechanical level, carding involves separating clumps of wool by pulling individual fibres out of them with the hooks on the surface of the workers. The collecting power of the workers and doffer is in part related to the probability that a fibre will be hooked by the worker, but there is another factor that contributes—the strength of the fibres and clumps.

The withdrawal force of a fibre is the force that must be applied to a fibre in order to remove it by its end from a clump of wool. The magnitude of this force depends on the number of fibre-to-fibre contacts along the trapped end of the fibre, and the friction force at each contact point. Studies of the withdrawal force by Grosberg and Smith [31, 32] have shown that it is proportional to the pressure applied to the assembly, but that even at zero applied pressure there is an intrinsic withdrawal force F_{W0} caused by the inherent entanglement of the fibres. Grosberg [31] gives the equation for the withdrawal force F_W per unit length of a single fibre from an untwisted sliver as

$$F_W = \mu' P + F_{W0}, \tag{2.4}$$

where P is the external pressure exerted on the sliver, and F_{W0} is the value of F_W at $P = 0$. $\mu' = \frac{2\mu d}{\phi}$ is a proportionality factor (with dimension length), in which μ is the coefficient of friction, d is the mean fibre diameter, and ϕ is the packing fraction in the sliver.

Equation (2.4) assumes that the friction force at each contact point is the same. In reality this is not the case. A contact point forms when the fibre wraps around a neighbouring fibre. The angle about which the fibre wraps around the contact point defines the friction force, by way of the capstan equation [33],

$$T_{load} = T_{hold}e^{\mu\phi}, \tag{2.5}$$

where T_{load} is the applied tension to the fibre, T_{hold} is the resulting force exerted on the other side of the contact point, μ is the coefficient of friction between the fibres, and ϕ is the total angle swept out by all turns of the fibre.

Figure 2.6 shows how eq. (2.5) can be applied to a fibre. T_{load} at the contact point closest to the pulled end of the fibre is the withdrawal force F_W, and T_{hold} at that point is equal to T_{load} at the next point. This continues along the fibre until the second-to-last contact point, where T_{hold} is equal to the static friction force between the fibre and the final contact point (which it only contacts tangentially). If a fibre has n contact points, then the withdrawal force will be

$$F_W = F_n \prod_{i=1}^{n-1} e^{\mu\phi_i}, \tag{2.6}$$

where F_n is the static friction force between the fibre and the nth (final) contact point, and ϕ_i is the total angle swept out by all turns of the fibre at the ith contact point. F_n will depend on the bending strain within the fibre around contact point $n-1$, as well as the distance between $n-1$ and n.

Another point that is not obvious from eq. (2.4) is that the withdrawal force depends on which end of the fibre is being pulled. As mentioned in section 1.1, the high felting ability of wool comes from the ability of wool fibres to ratchet through a fibrous assembly in the direction of the root. The ratcheting mechanism arises from anisotropy in the frictional force of the fibre surface. The cuticular cells on a wool fibre have a tooth-like shape, which causes the coefficient of friction in the direction away from the root to be lower. In its natural environment where the fibre is immobile, this acts as a self-cleaning mechanism; dirt particles will migrate out of a sheep's fleece because less energy is required to overcome static friction. In wool products however, the fibres themselves are mobile, and will move in the direction of their root for the same reason.

Finally, eq. (2.4) also assumes that no additional contact points are made as the fibre is withdrawn; this is unlikely to be true in practice, at least for products in their initial unfelted state. Recently, Lee et al. [34] developed a model for describing the forces acting on a single fibre as it is withdrawn from a tangled fibre assembly. They defined the "effective length" of the fibre as the length over which the friction forces act, and explained that this will first increase and then decrease as the fibre is withdrawn. Figure 2.7 shows a schematic representation of a fibre being withdrawn from a fixed tuft structure. As the fibre begins to move, its trajectory converges on the shortest trajectory and the fibre comes into active contact with neighbouring fibres. It is these fibres that dominate the withdrawal force, and their number increases as the "slack" in the fibre is taken up. Once the entire fibre is following the shortest trajectory, continued withdrawal decreases the effective length as the total length of fibre remaining in the tuft decreases.

2.3.2.1 The effect of PLA on fibre migration

Fibre migration occurs when fibres incur transient tension due to movement or agitation of the assembly. This is functionally equivalent to applying a force to the end of a fibre (or fibre segment). If this transient force exceeds the withdrawal force for the fibre (or fibre segment), the fibre will migrate. In section 1.2, it was hypothesised that one of the reasons for the observed performance benefits of including PLA fibres in knoppy web was that they reduce fibre migration. We now use eq. (2.6) to propose a mechanism by which this could occur.

In a pure wool batt, the contact points are not fixed in place and can be easily deflected. The tension in the fibre creates a lateral force on the contact points, from which the frictional force arises; however, this will also "push" the contacting fibres sideways. This has the effect of "straightening" the fibre's path, reducing ϕ_i at each deflected contact point, which in turn decreases the capstan force at those points.

In a bonded knoppy web, the PLA substructure has two effects:

- It fixes the position of the PLA fibres. This inhibits their deflection, meaning that a larger lateral force is required to deflect the PLA fibres.
- It limits the potential range of deflection of wool fibres. In a well-blended fibrous assembly, the wool fibres will be effectively interwoven throughout the bonded PLA substructure. Deflected wool fibres would therefore be expected to quickly come into contact with nearby PLA fibres.

Thus, the path of wool fibres under tension will undergo far less "straightening" in the presence of the PLA; this lessens the reduction of ϕ_i and the capstan force at each contact point. By eq. (2.6), the nominal threshold to fibre migration will therefore be higher in a batt containing PLA fibres than in a pure wool batt. This would directly lead to the observations that knoppy web exhibits less clumping and felting than traditional wool batting.

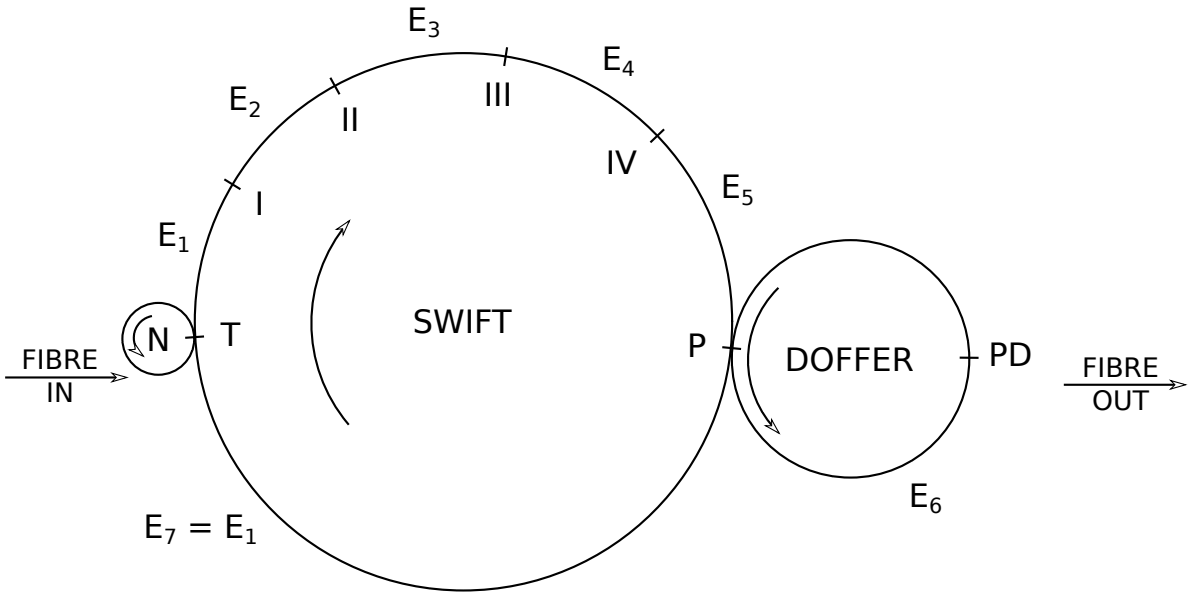


Figure 2.5: States to be considered in the progress of the fibre through the card, with 1 swift and 4 carding points (I to IV). (The workers and the strippers are not shown in the figure) [30]

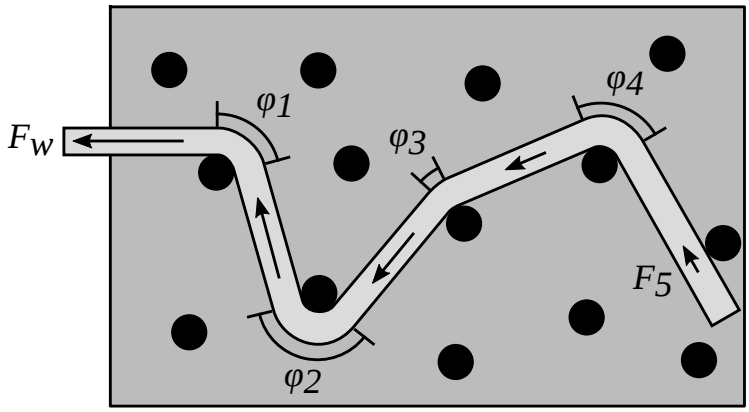


Figure 2.6: Constructing the fibre withdrawal force from capstan forces.

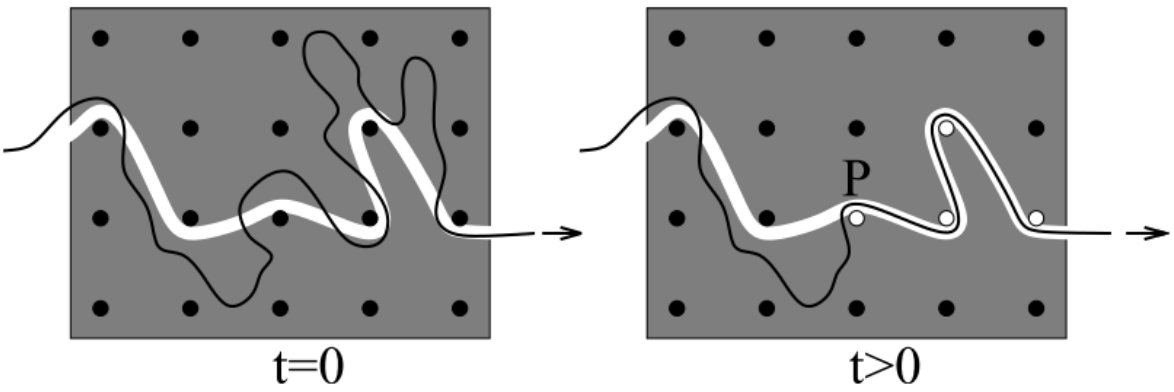


Figure 2.7: Schematic illustration of a fibre being removed from a tuft. The unshaded line indicates the shortest trajectory for the fibre's withdrawal when constrained by neighbouring fibres represented here by dots [34].

We believe that the work of Lee et al. [34] is a rich area for new physics, because the fibre migration observed with progressive washing could potentially be described as an aggregate of fibre withdrawal processes. The mechanism described above could form the basis of a theory describing the micromechanics of washing fibrous assemblies. A similar idea could also be applied to the presence of knops in the knoppy web; their geometric constricting effect on their constituent fibres could similarly increase the fibre withdrawal force. Development of these ideas would be an ideal excellent potential basis for future work.

2.3.3 Fibre breakage

In section 2.3 we mentioned that the collecting power of a worker has a clear dependence on the worker speed. Increasing the worker speed therefore is an effective way to increase the level of carding and mixing. But it causes other problems, foremost of which is an increase in fibre breakage.

Carnaby and Wood [36, 37] examined the physical configuration required to cause a fibre break, and developed a transformation between the intial and final length distributions based on the work of Meyer et al. [38]. Figure 2.8a shows the geometry: a carding element (such as a tooth on the swift) gripping a fibre withing a clump. The fast-moving carding element (relative to the clump) applies a force to the fibre. This force must be strong enough to overcome the withdrawal force for the short tail of the fibre in order to card it. However, if the required withdrawal force is larger than the breaking load, the fibre breaks.

Carnaby and Wood only considered breakages at the tooth, because this is where the majority of breaks occur [36]. However, in tender wool this is not necessarily the case—certain regions of the fibre may be weaker than others. Carnaby and Burling-Claridge [35, 39] accounted for this by replacing the length-only distribution with a joint length-area distribution, representing the variable fibre strength as a non-uniform cross-sectional area. Geometrically, the situation is unchanged (fig. 2.8a), but the fibre now breaks if the tension at any point along the fibre (fig. 2.8b) exceeds the local fibre strength.

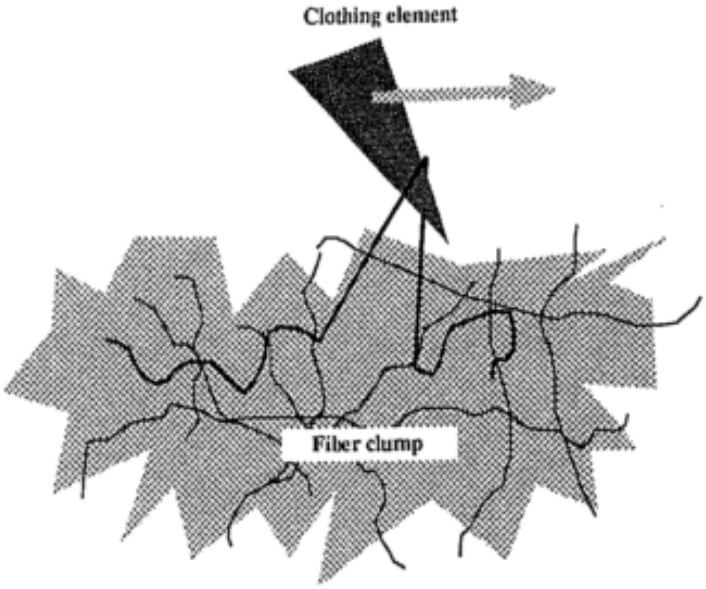
Breakages during carding (or at other stages in processing) will shift the length distribution of the fibre stock towards shorter lengths. Several authors have examined the effect of carding fibre breakages on fibre length distributions, and section 2.4.1 covers this in more detail.

2.4 Cutting

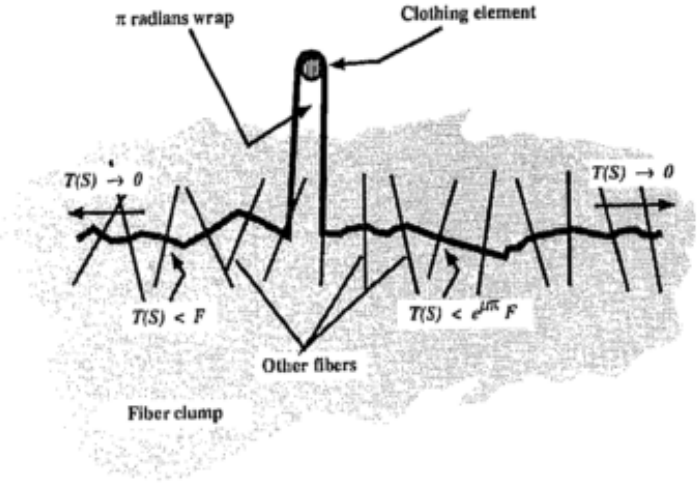
As convenient as it would be for wool fibre to be available in standard lengths, this is a luxury reserved for synthetic fibres. Between natural growth and variations in shearing, as well as fibre breakage during processing, bulk wool fibre has a length distribution which will vary from batch to batch. Reliably determining this distribution is trivial in a laboratory setting [40]; the current standard measurement instrument is the Almeter, which uses a capacitance method to determine length distributions [41]. Expensive dedicated testing equipment is not however a justifiable expense for most companies, and is impractical in most factory environments. A number of indices have been proposed over the years for characterising fibre length distributions via simple and cheap tests (e.g. [42]), but these only provide indications of relative differences. As it is, most commercial production lines do not have the time or resources to be able to test every batch of bales they receive.

Even when the characteristic length of a wool batch is suitable for a product, the fibre length distribution can sometimes span several centimetres [43]. Wider distributions will have non-trivial fractions of longer and/or shorter fibres, which can have a noticeable effect on the form and quality of the product. This is especially true for the knopping process; as outlined in section 2.6, the presence of longer fibres can cause knops to be less well-formed, which is detrimental to the properties of knoppy web.

Due to these two issues, it is usually necessary to shorten the fibres in its product blends; this is always performed on the carded sliver, when the fibres are in a manageable state. There are two common



(a) Uncarded fibre in a clump, in contact with a carding element [35].



(b) Fibre forces and tensions when in contact with the carding element [35].

Figure 2.8

methods for shortening fibres:

- *Cutting*—the sliver is passed through a cutting knife, which slices the sliver into segments.
- *Stretch-breaking*—the sliver is passed through two pairs of rollers, spaced a distance r apart. The front rollers run at a higher speed than the back rollers; this causes any fibres with a length greater than r to be broken (when they are clamped by both rollers at once), while shorter fibres are left untouched.

Cutting a sliver generates a larger fraction of short fibres than stretch-breaking, because fibres shorter than the cutting length can still be cut, whereas fibres shorter than r will never be broken. However, cutting performs better at shortening fibres than stretch-breaking when the fibres in the card sliver are not parallel [44]. This makes cutting a preferable option for production lines without gilling facilities [28] when long fibres are detrimental to the end product (such as knoppy web).

2.4.1 Transforming a fibre length distribution

Cutting the sliver does not result in wool stock with a uniform length. While some fraction of the long fibres will be trimmed to the cut interval, a larger fraction will be sliced into two smaller segments. Fundamentally, the cutting process is shifting the length distribution of the wool towards shorter lengths; this process can be treated as a transformation.

The standard technique developed by Lindsley for measuring fibre orientation in slivers [45] involves several cutting operations. Tallant and Pittman [46] used the Lindsley combing ratio to derive the transformation equation for the normalised probability density function $f(x)$ of the length of fibres lying at random in a sliver being cut at intervals t:

$$g(x) = \frac{t}{t+m}\left[\left\{f(x)\left[1-\frac{x}{t}\right]+\frac{2}{t}\left[1-F(x)\right]\right\}\left\{1-H(x-t)\right\}+N(t)\delta(x-t)\right], \tag{2.7}$$

where $F(x)$ is the cumulative probability function of the uncut fibres, m is the mean fibre length, H is the Heaviside step function, δ is the Dirac delta function, and

$$N(t) = F(t) - 1 + \frac{1}{t}\int_t^\infty x' f(x')\,\mathrm{d}x'. \tag{2.8}$$

Meyer et al. [38] took an alternative statistical approach, and derived a general equation for transforming a fibre-length distribution via a uniform fissioning process (such as breaking or cutting):

$$N_2(L)\,\mathrm{d}L = N_1(L)\,\mathrm{d}L\,(1-q(L)) + \int_L^\infty N_1(x)q(x)n(x)S(x,L)\,\mathrm{d}L\,\mathrm{d}x \tag{2.9}$$

where $N_1(L)\,\mathrm{d}L$ is the original length distribution, $q(x)$ is the fraction of fibres originally of length x that are fragmented, $n(x)$ is the number of fragments into which a fibre originally of length x is fragmented, and $S(x,L)\,\mathrm{d}L$ is the fraction of fragments from a fibre originally of length x which are of length $x' \in [L, L+\mathrm{d}L)$ after fragmenting.

Burling-Claridge and Carnaby [47] used eq. (2.9) as the basis for their theoretical analysis of cutting a sliver into a series of sections of length $C+\mathrm{d}C$, where $\frac{\mathrm{d}C}{C} \ll 1$. They assumed that all fibres were aligned perpendicular to the cutting direction, and that all fibres were straight i.e. crimp was ignored. Using the geometry shown in fig. 2.9, they evaluated the various parameters of eq. (2.9) for various combinations

of x and y, and arrived at the transformation equation

$$N_2(L)\,\mathrm{d}L = \begin{cases} 0 & \text{when } L > C, \\ \displaystyle\sum_{j=1}^{\infty} \int_{jC}^{(j+1)C} N_1(x) \frac{(C+x)[(j+1)(j-2)C+2x]}{(j+1)(j+2)C^2}\,\mathrm{d}L\,\mathrm{d}x & \text{when } L = C, \\ \displaystyle N_1(L)\,\mathrm{d}L\,(1-\frac{L}{C}) + \int_{L}^{C} N_1(x)\frac{C+x}{C^2}\,\mathrm{d}L\,\mathrm{d}x & \\ \displaystyle +\sum_{j=1}^{\infty}\left(\int_{jC}^{jC+L} N_1(x)\frac{2(C+x)}{(j+1)C^2}\,\mathrm{d}L\,\mathrm{d}x\right. & \text{when } L < C. \\ \displaystyle \left. \quad + \int_{jC+L}^{(j+1)C} N_1(x)\frac{1(C+x)}{(j+2)C^2}\,\mathrm{d}L\,\mathrm{d}x\right) & \end{cases} \tag{2.10}$$

Under their set of assumptions, the transformation of a fibre-length distribution by cutting depends only on the cutting length C.

Burling-Claridge and Carnaby verified their transformation equation by measuring the fibre-length distribution of a sliver before and after cutting, and comparing the measured cut distribution to the predicted distribution. Figure 2.10 shows both distributions, while fig. 2.11a(ii) show the fibre length distribution predicted by eq. (2.7) for cutting an ideal sliver with originally-uniform fibre length (fig. 2.11a(i)).

It is clear that eq. (2.10) provides a good prediction of the mean hauteur (fibre length biased by fibre cross-section). Nevertheless, there are two clear discrepancies between the prediction and reality:

- The experimental distribution extends beyond the cut length.
- There is an additional peak (beyond the predicted maximum) at very short fibre lengths.

The first discrepancy is an obvious consequence of the two assumptions made during the derivation. The second discrepancy can be explained by considering another factor ignored in the analysis: the presence of hooked fibres (see section 2.4.2).

The assumption that all fibres are aligned perpendicular to the cutting direction is in practice false (except in gilled slivers [28]). Fibre orientation in a carded sliver is readily characterised using the Lindsley technique [45], which defines an "orientation index" to express the degree of orientation of fibres in a sliver. A perfectly-aligned sliver would have an orientation index of 100, while slivers with non-parallel fibres would have lower indices. Typical values for a carded sliver are around 87–90 [45]. Thus the carding process is not sufficient to produce a well-aligned sliver; without additional processing (such as gilling [28]) there will be a non-trivial fraction of fibres that lie at an angle to the axis of travel. These fibres will have a smaller "extent" in the sliver (being the length of the fibre's projection onto the axis of travel), and consequentially will have a smaller probability of being cut. In terms of eq. (2.9), $q(x)$ decreases as the orientation index decreases, and therefore the fraction of longer fibres in the final distribution increases.

Harrowfield [44] extended eq. (2.7) to handle disoriented slivers, obtaining the transformation equation

$$\left(1+\frac{m\langle\cos\theta\rangle}{t}\right) g(x) = \begin{cases} \displaystyle f(x)\left[1-\frac{x\langle\cos\theta\rangle}{t}\right] + \frac{2\langle\cos\theta\rangle}{t}\left[1-F(x)\right] & \text{when } x < t, \\ \displaystyle f(x)\left\{2\int_{\cos^{-1}(t/x)}^{\pi/2}\left[1-\frac{x}{t}\cos\theta\right]\psi(\theta)\,\mathrm{d}\theta\right\} & \\ \displaystyle +\frac{4}{t}[1-F(x)]\int_{\cos^{-1}(t/x)}^{\pi/2}\cos\theta\psi(\theta)\,\mathrm{d}\theta & \text{when } x > t, \\ \displaystyle +\frac{2t\psi[\cos^{-1}(t/x)]}{x\sqrt{x^2-t^2}}\left[F(x)-1+\frac{1}{x}\int_x^{\infty} x' f(x')\,\mathrm{d}x'\right] & \end{cases} \tag{2.11}$$

where $\psi(\theta)$ is the angular distribution function, and

$$\langle\cos\theta\rangle = 2\int_0^{\pi/2}\cos\theta\psi(\theta)\,\mathrm{d}\theta \tag{2.12}$$

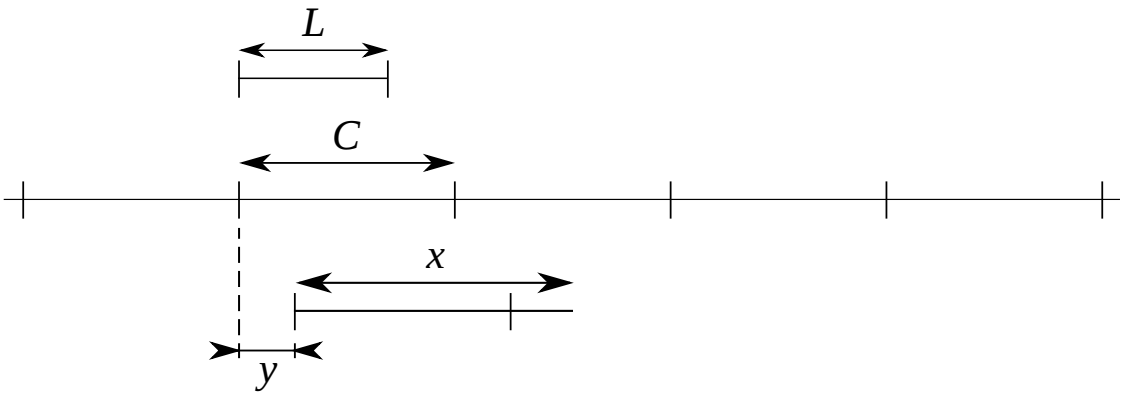


Figure 2.9: Geometry for calculating length fractions of cut fibres [47]

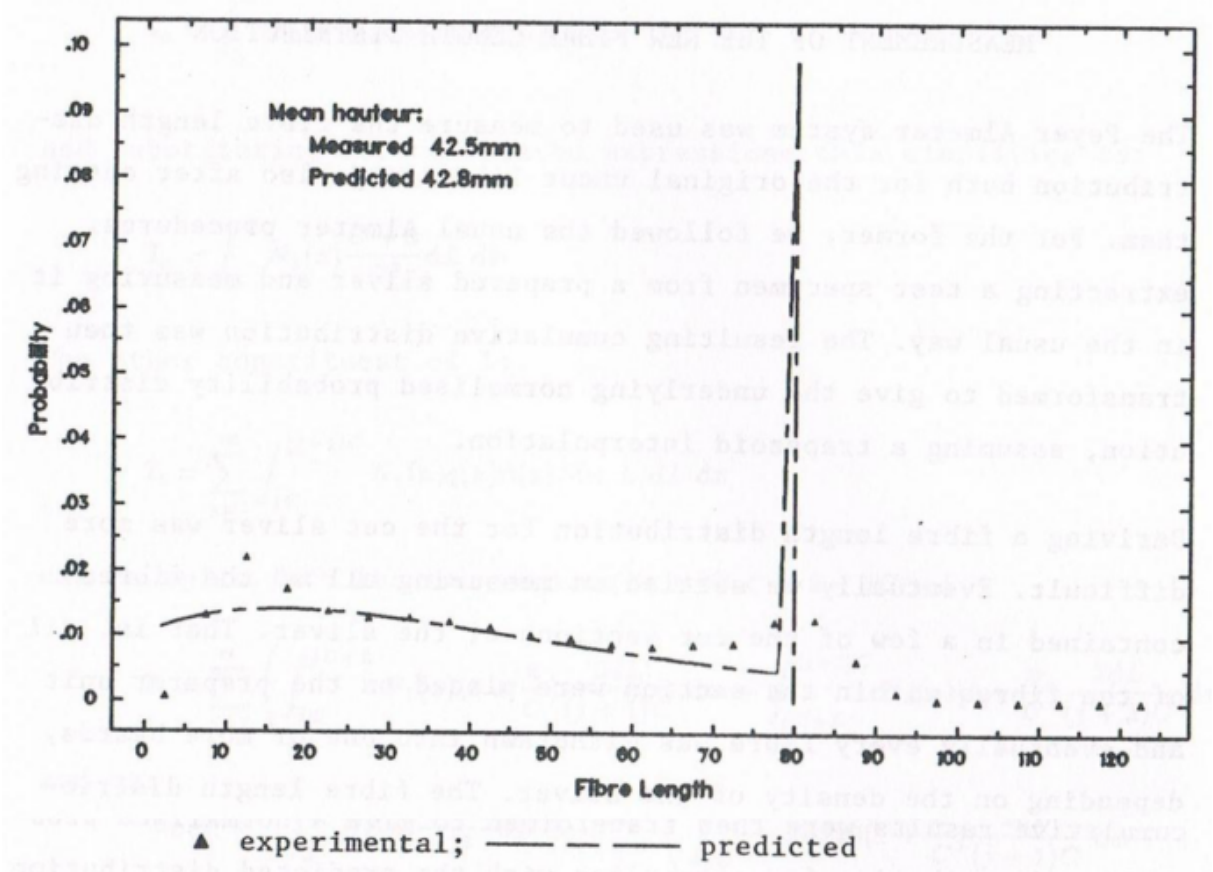
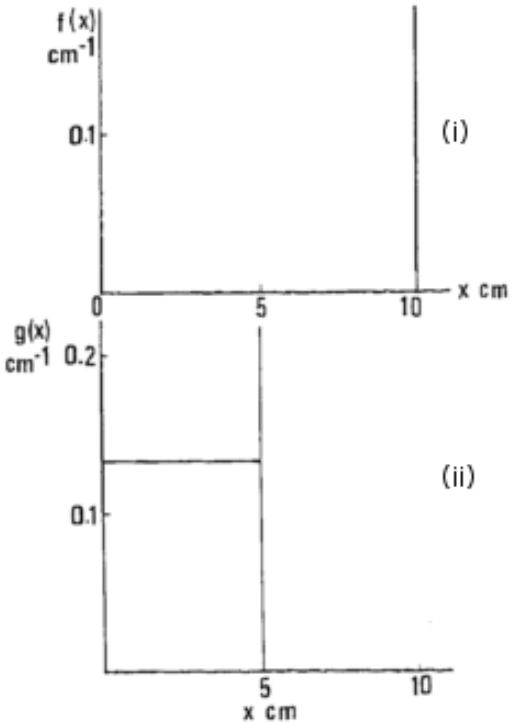
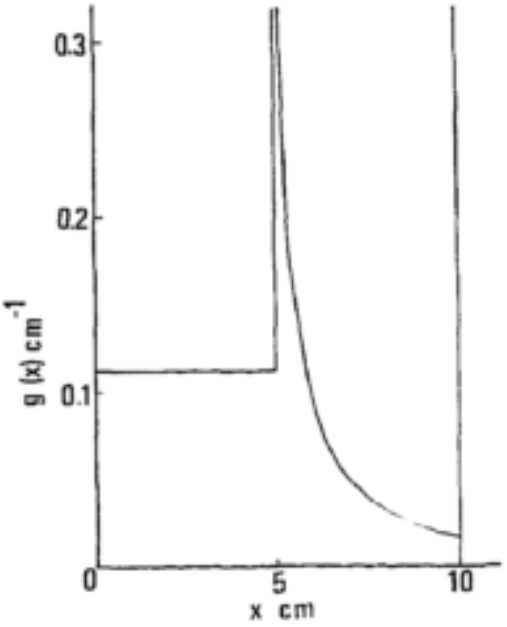


Figure 2.10: Comparison of distributions for a sound crossbred wool of medium length (barbe = 100 mm) [47].



(a) Probability density of (i) parent fibres all 10 cm in length; and (ii) the daughter that results from systematic cutting at 5 cm intervals of a parallel sliver [44].



(b) Daughter probability density that results from the systematic cutting at 5 cm intervals of parent sliver in which fibres of length 10 cm are randomly disoriented [44].

Figure 2.11

is the average value of $\cos\theta$. Figure 2.11b show the fibre length distributions predicted by eq. (2.11) for cutting the distribution shown in fig. 2.11a(i); there is a visible tail above the cutting length similar to the experimentally-measured distribution in fig. 2.10.

The assumption that crimp can be ignored has a similar effect to the fibres not being parallel. The inherent crimp of wool fibres may get stretched out during processing steps such as carding[3], but afterwards will undergo relaxation. This means that even in a well-aligned sliver, the extent of the fibre along the axis of travel will in general be shorter than the true fibre length, and thus fewer fibres will be cut.

2.4.2 The effect of hooked fibres in a sliver

As described in section 2.3, the surfaces of the rollers in a card are covered in wire teeth, which pick up and transfer fibres. It comes as no surprise then that most fibres in the final sliver will be "hooked," i.e. the fibre will contain one or more 180° bends.

Morton and Summers [48] were the first to show (via tracer-fibre studies) that the majority of fibres in a carded sliver are hooked, with only 20% of the tracer fibres they observed having no hooks. They classified the hooks as "leading" or "trailing" based on whether the hook precedes or follows the main fibre body out of the card, and observed that trailing hooks are more numerous than leading hooks, outnumbering them 3 to 1 [48, 49]. Wakanar et al. [49] examined how different processing conditions affected the formation of hooks, and found that the primary driver of hook formation is the card's rate of production or cylinder loading. Other carding parameters (such as the surface speeds of the cylinder or doffer, or the type of wire used) had no effect. The following observations were made:

- The number of leading hooks increased as the card's rate of production or cylinder loading was increased.
- At higher production or loading the number of trailing hooks decreased.
- At very high production rates, the ratio of leading to trailing hooks is almost 1.

The initial mechanism proposed for the formation of trailing hooks was that the leading end of a fibre is picked up by the doffer, and the hook forms by a fibre reversal when the trailing end is carried forwards by the faster swift [48]. However, the majority of trailing hooks are created by the combing effect of the swift. Fibres collected by the doffer are exposed to the fast-moving swift teeth while near the carding point, and will be combed downwards. Each fibre can therefore be considered to be hooked over a doffer tooth, with two "legs" combed down either side. The fibre has a much higher chance of staying on the doffer if the two legs are of similar length, so that the combing force is balanced on either side. Recalling that the doffer teeth point opposite to the direction of rotation, the majority of fibres collected by the doffer will therefore be trailing hooks with the hook in the middle.

Hooks contribute to the first discrepancy in fig. 2.10 in the same way as crimp and fibre orientation. Hooked fibres have a shorter extent, which lowers the probability of the fibre intersecting with the cutting edge. This increases the number of fibres longer than the cutting length in the final distribution. In fact, hooks are a significant contributor to the lack of a defined upper length limit: crimp can cause the extent of a fibre to be 20–30% less than its true length [48, 50], while a single trailing hook can intuitively cut the extent by up to 50%.

The second discrepancy in fig. 2.10 is the presence of an additional maximum at very short fibre lengths. Crimp and fibre orientation are insufficient to explain this, because these both have a lengthening effect on the distribution. However, hooks provide a ready explanation. Assume an otherwise straight fibre with a hook in its middle:

[3] The fibres are also stretched out during measurement, but the crimp is explicitly accounted for in the measurement procedure [45].

- When the cutting device slices the fibre near the free ends (fig. 2.12a), it will result in two short fragments and one long fragment. This single cut is equivalent to two cuts on an unhooked fibre at opposite ends of the fibre; both cuts are predicted statistically by eq. (2.10).
- When the cutting device slices the fibre near the hooked end (fig. 2.12b), it will result in one short fragment and two longer fragments. This single cut is equivalent to two closely-spaced cuts in the middle of a single unhooked fibre. Equation (2.10) predicts statistically that there will only ever be a single cut at the middle of a fibre, and thus misses this additional short fragment.

Therefore, the cut distribution of slivers containing hooked fibres will contain a higher fraction of short fragment lengths than is predicted by eq. (2.10), as was observed.

2.5 Blending

A key aspect of manufacturing a product from multiple fibre types and components is blending. One of the hypothesised benefits of including PLA in the knoppy web is that it forces fibre separation, inhibiting felting of the wool (hypothesis 4). For it to do this, however, the PLA must be well-dispersed within the knoppy web. There is little point in producing a web that contains clumps of PLA and regions of pure wool; under bonding the PLA will form discrete rigid blobs that will decrease the quality of "feel" of the end product, while under repeated use the wool regions will undesirably felt and lead to a decrease in bulk. Likewise, an even distribution of knops throughout the knoppy web is essential to ensure that the overall properties of the knoppy web are consistent and uniform; regions with a lower density of knops can have lower resilience to compression, and an uneven knoppy web will have poorer thermal performance. Therefore, it is important to examine practical strategies for fibre and component blending, and balance the technical capabilities of the blending process with the industry requirements on time, cost and the overall value added to the end product.

2.5.1 Carding

Carding is the primary tool that can be used to blend fibres. Section 2.3.1 showed that carding is an effective microscale fibre blending technique: due to the expansion of the fibre stock on the swift into a very thin layer, it is very easy for different fibre types to become layered together and subsequently intermingled in the output sliver. Carding also has a blending effect at mesoscale, because the workers take a fraction of the fibre on the swift and return it some distance behind. Of interest now is the macroscale blending power: how far along the sliver is the original fibre cluster spread?

If there were no workers, and the doffer had a collecting power of 1, then each fibre would travel the minimum possible distance through the card—the distance from T to P (fig. 2.5). In this configuration there is no mixing, and the relative positions of the fibres is essentially unchanged by the card. However, every time a fibre is collected by a worker, or escapes the doffer, it must travel an additional "loop" before it has another chance to exit the card. This delays the position of the fibre in the sliver relative to its uncarded position. The average displacement of a fibre from its uncarded position can be used as a measure of the post-carding extent of a fibre cluster, and therefore of the macroscale blending power of the card.

Recall that eq. (2.3) gives the mean time that a fibre spends inside the card. We can write the minimum time that a fibre spends inside the card as

$$\theta_{\min} = \frac{\alpha}{\omega_t}, \tag{2.13}$$

and subtracting the two, we can define the mean delay θ_D as

$$\theta_D = \frac{1}{p_d}\left(\frac{q_d}{\omega_t} + \sum_1^4 \frac{p_i}{q_i \omega_i'}\right). \tag{2.14}$$

Therefore, we can write the average displacement of a fibre in the sliver from its uncarded position as

$$\begin{aligned}\delta_f &= \pi r_d^2 \omega_d \theta_D \\ &= \frac{\pi r_d^2 \omega_d}{p_d}\left(\frac{q_d}{\omega_t} + \sum_1^4 \frac{p_i}{q_i \omega_i'}\right),\end{aligned} \tag{2.15}$$

where r_d is the radius of the doffer, and ω_d is the angular velocity of the doffer.

Figure 2.13 shows displacement of a fibre from its uncarded position in the sliver as a function of the collecting power of the workers and doffer. The two expected trends are present:

- Displacement increases as the collecting power of the workers increases.
- Displacement decreases as the collecting power of the doffer increases.

The collecting power of the workers and doffer does have a dependence on their angular velocity (as mentioned in section 2.3), but from eq. (2.15) changes in angular velocity will simply have a scaling effect on fig. 2.13.

For mid-range values of $p_w = 0.5$ and $p_d = 0.5$, we see that $\delta_f \approx 4$m. This shows that the card is providing good medium-range blending of fibres. In the most extreme carding situations ($p_w = 0.8$ and $p_d = 0.2$), δ_f reaches about 40m. However, an average 100kg fibre stock can card into 4000 m of 25 kilotex sliver, so even with maximal carding, variations in the fibre stream to the card with a long period can still manifest in the carded sliver. Thus a card alone is insufficient; other blending tools are necessary so that the fibre stream to the card is roughly uniform on a macroscopic scale.

2.5.2 Opening

Another tool that fibre production lines have at their disposal for blending is a bale opener (fig. 2.14). Wool bales are packed very densely to minimise transport costs; an average 50 kg wool bale may have been compressed to as little as 10% of the wool's original volume. The opener reverses this compression, teasing apart the staples and restoring bulk. The fibre output from the opener is blown into a silo. A spinning funnel at the top of the silo (fig. 2.15) distributes the fibres evenly across the silo floor.

Unlike a card, an opener does not have much inherent blending power; its purpose is simply expansion of the fibre stock. However, the initial conveyor belt that feeds the opener can be used to provide macro-blending. The horizontal conveyor belt leads to a near-vertical toothed feed sheet; by layering the different fibre types in the feed hopper (as in fig. 2.16), the feed sheet tends to pick up a mixture of fibre types.

The opener is designed to take a single bale at a time, and therefore only provides adequate blending on small-scale production runs (where the total fibre weight to be blended is no more than 40 kg to 50 kg). The purpose of the silo after the opener is to assist with blending fibres across the individual bales being used in a particular production run. A 50 kg bale at 10:1 compression becomes a vertical "slice" about 15 cm to 20 cm thick in the cylindrical silo after passing through the opener. The fibre is subsequently sucked out of the silo from the side for later processing. Much like the opener's feed sheet, this ensures that the fibre stream to the next processing step contains fibres from multiple bales; thus the silo helps to scale the opener blending to larger production runs.

The silo also helps to defend against blending issues caused by differences in fineness of the fibre components. The toothed feed sheet of the opener can struggle to pick up lighter and finer fibres, such as PLA (particularly if the PLA has been pre-opened via carding). This can cause a variation in the fibre ratio of the feed to the opener: the fibre entering the opener will be initially wool-rich, and later will be PLA-rich. In the silo, this translates to a vertical fibre blend gradient. By then sucking out the fibres from the silo sideways, the gradient in the fibre blend is neutralised to some extent.

(a) (b)

Figure 2.12: Geometry for calculating length fractions of cut hooked fibres

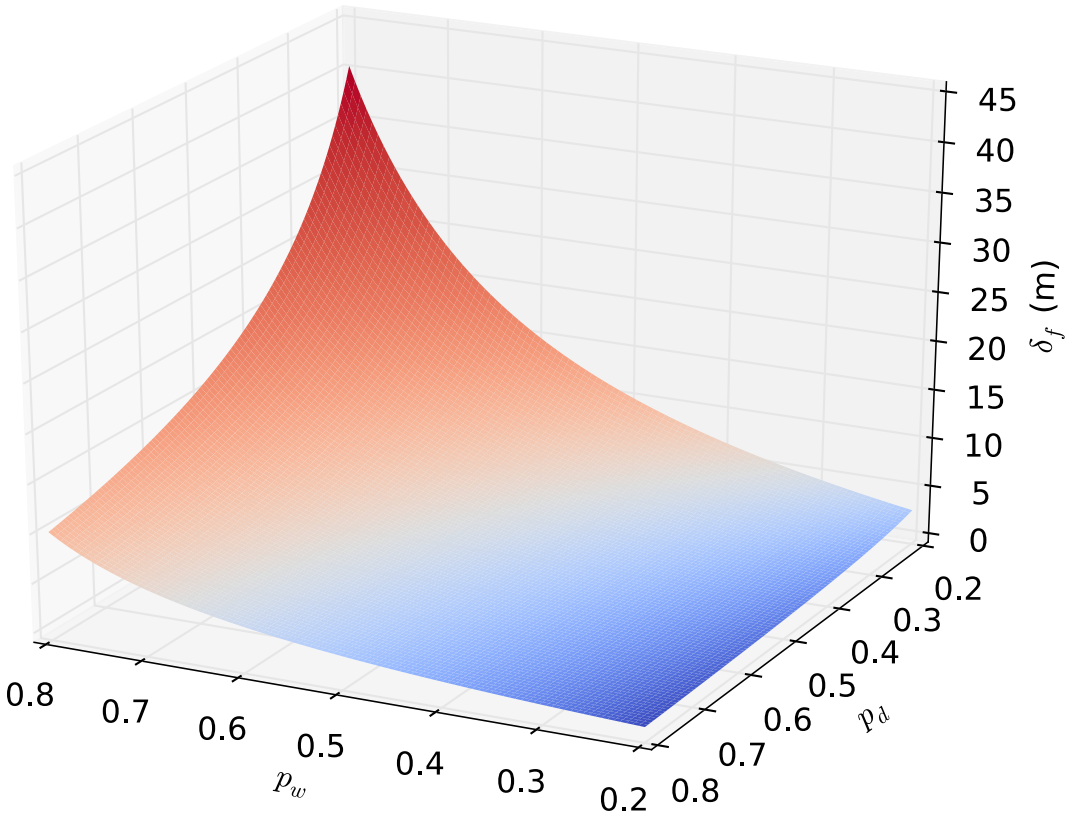


Figure 2.13: The average displacement of a fibre in a sliver as a function of the collecting power of the workers (p_w) and the doffer (p_d). ($\omega_i' = 10\,\mathrm{s}^{-1}$, $\omega_t = 200\,\mathrm{s}^{-1}$, $\omega_d = 10\,\mathrm{s}^{-1}$, $r_d = 0.4\,\mathrm{m}$.)

Figure 2.14: The feedsheet of the opener.

Figure 2.15: The spinner that distributes the opened fibres through the silo.

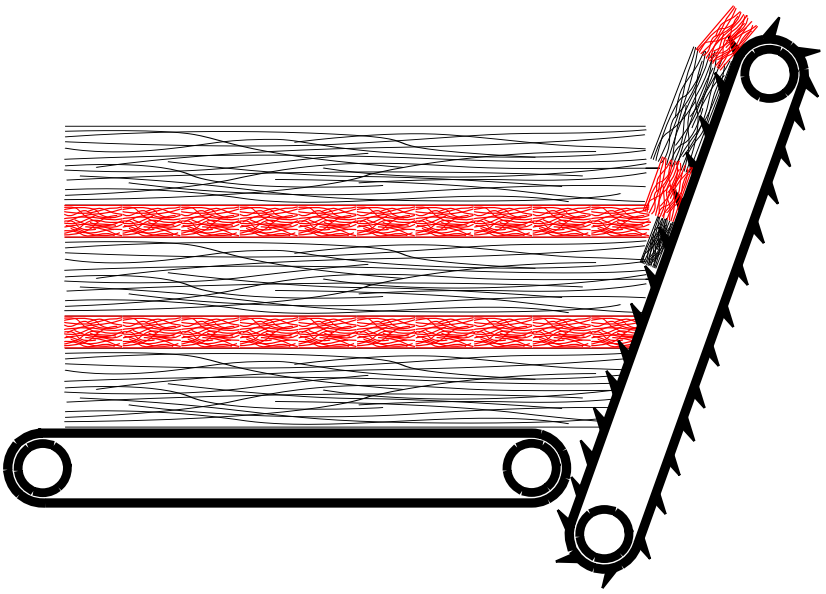


Figure 2.16: Schematic showing how the feed conveyor belt on the opener can be used to blend fibres at small scales.

2.5.3 Pre-melling

Pre-melling is the process of blending two or more fibre components together in a series of blending steps, to increase the overall efficiency of the blend. The primary use for pre-melling is in situations where one fibre component greatly outweighs the other components, or where the properties of the fibre components differ considerably. If only one blending step is used, the end product can end up with small "pockets" or clumps of like fibres, negating the benefit of having the multiple fibre components. Instead, fractions of the fibre components are blended together, and then these blends are themselves blended with more of the fibre stock (or other pre-blends).

Pre-melling can occur at several stages of the production process:

- *Carding*—sending the fibre blend through multiple carding steps provides more chances for fibre tufts to be separated and mixed, at the expense of increased fibre breakage (section 2.3.3) and loss.
- *Opening*—multiple passes through the opener can be used to initially blend smaller quantities of fibre stock together, and then subsequently blend them into larger fibre stock. The combination of an opener and silo can also be considered to be pre-melling.
- *Drafting*—in production lines with drafting capabilities [51], multiple carded slivers can be drafted into a single sliver.

Pre-melling via drafting is known as "doubling," and is a particularly effective technique for both blending fibres and reducing sliver irregularities. Multiple slivers are fed into the drafter, and the drafter then attenuates the slivers by a draft equal to the number of slivers, giving a single output sliver of similar linear density to the input slivers. The multiple slivers are usually created during the same carding run; thus the doubling process blends fibre together that may have exited the card hundreds of metres apart, averaging out any time-dependent irregularities.

The blending power of doubling comes from the progressive mixing of "ribbons" of sliver. Figure 2.17 illustrates this process visually for two fibre types (colours). If eight slivers are passed into a drafter with a set draft of eight, then the output sliver will consist of eight ribbons, each one eighth of their original linear density. If the doubling is repeated, there will be sixty four ribbons, each even finer. In general, the number of ribbons in the final sliver is

$$N_r = S^R, \tag{2.16}$$

where S is the number of slivers used as input to each doubling step, and R is the number of doubling steps. For $S = 8$, fig. 2.17 shows that $R = 3$ gives a reasonable level of blending.

Martindale [52] showed that for linear products (such as yarns or slivers) with a random arrangement of fibres and average number of fibres per cross-section n, the coefficient of variation of cross-sectional area V_a is

$$V_a = \frac{100\sqrt{1 + 0.0004V_D^2}}{\sqrt{n}}, \tag{2.17}$$

where V_D is the coefficient of variation of the fibre diameter. Fibre diameter is an inherent property of a fibre blend, and the equation contains no dependence on fibre length or any other property that can be controlled at processing time. Thus V_a represents the minimum level of irregularity that can possibly be achieved, no matter how many doubling steps are performed.

2.5.4 Cutting

As strange as it may sound, the sliver cutter can also be a useful tool for blending fibres. Section 2.5.2 described how the feed hopper of the opener can be used for small-scale blending of fibre components. This method is limited by the size of the hopper, and is time-consuming. However, if the fibre components need to be cut prior to being blended together, then two steps can be combined into one.

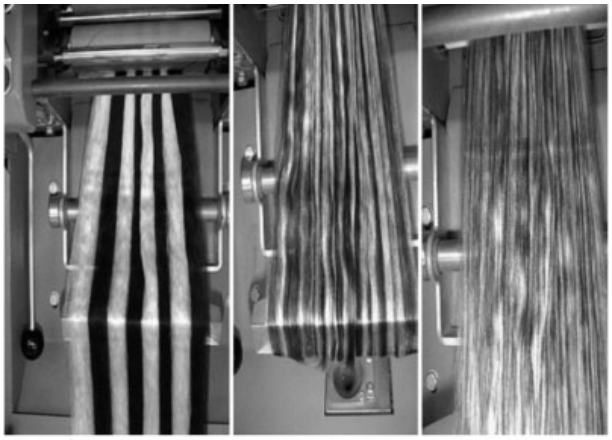


Figure 2.17: Drawframe passages with eight-sliver feed and a draft of eight. Example of blending by doubling and drafting: drawframe blending [28].

Say, for example, that two carded slivers (A and B) needed to be cut and then blended together at a ratio of 75:25. By the method in section 2.5.2, each sliver would be cut separately, and then the cut fibres would be layered into the hopper in a roughly 3-to-l ratio.

Alternatively, the slivers could be cut at the same time, with 3 slivers of A entering the cutter alongside 1 sliver of B. This removes a step (the combined cut fibre can simply be dumped into the feed hopper), and in fact enables the opener-based blending step to be scaled to larger production runs. The resulting cut fibre is also far more intimately mixed than could feasibly be achieved by hand-layering at the opener. Functionally, this is equivalent to the doubling process described in the previous section—and in fact, doubling itself is often used during the cutting process to improve blending.

In cases where the cutting step is used as part of pre-melling, the pre-melling ratios should be chosen to result in an integer ratio of slivers. If the production run is a repeat, then it is also possible to factor in any difference in linear densities of the slivers.

2.5.5 Number ratio of different fibre types

One of the main reasons for blending is to ensure that the various fibre components are intimately mixed. The efficacy of different blending strategies depends in part on the relative number of fibres of each component (the number ratio). It is generally easier to evenly blend equal numbers of fibres than to blend a small quantity of one fibre type into a large quantity of another.

Let there be an assembly of fibres of circular cross-section of type f with total mass m_f. The number N_f of fibres in that assembly is

$$N_f = \frac{4}{\pi} \frac{m_f}{\rho_f l_f D_f^2}, \tag{2.18}$$

where D_f is the fibre diameter, l_f is the mean fibre length and ρ_f is the fibre density. If we blend these into a mass $m_{f'}$ of fibre type f' (also of circular cross-section), then we can write the number ratio of the two fibre types as

$$\frac{N_f}{N_{f'}} = \frac{m_f}{m_{f'}} \frac{\rho_{f'}}{\rho_f} \frac{l_{f'}}{l_f} \left(\frac{D_{f'}}{D_f} \right)^2. \tag{2.19}$$

The number ratio $N_f/N_{f'}$ is proportional to the mass ratio $m_f/m_{f'}$ as expected, but inversely proportional to the density and length ratios, and inversely proportional to the square of the diameter ratio.

Synthetic fibres usually have a specified linear density instead of a diameter, often given in units of Denier. The conversion between Denier and fibre diameter for a fibre of circular cross-section is

$$D_f = \sqrt{\frac{4000}{9\pi} \frac{\text{Denier}}{\rho_f}}, \tag{2.20}$$

where D_f is in µm and ρ_f is in $\mathrm{g\,cm^{-3}}$.

Table 2.2 shows the number ratios for various wool:PLA blends, using the fibre properties given in table 2.1. A density value was not provided for the PLA fibres, however the material density range has no effect on the final number ratio, because $\rho_{f'}$ in eq. (2.19) cancels out with ρ_f in eq. (2.20).

It is clear that both fibre length and fibre diameter have a significant effect on the number ratio. The PLA–51 blend contains similar-length fibres, but the PLA fibres are only two-thirds of the diameter of the wool fibres; this translates into a 55% reduction in the number ratio relative to the weight ratio. The PLA–32 fibre blends, with the additional short fibres, have a 72% reduction in the number ratio, with the PLA fibres becoming more numerous in the case of the 70:30 wool:PLA blend. Thus, even with a small fraction by weight of PLA fibres, they form a significant fraction of the fibre blend.

2.6 Knopping

Knopping is the key proprietary step in the production process. As outlined in section 1.2, the technology behind knopping has been developed over several decades, and at this stage its control parameters are

Table 2.1: Properties for some of the fibre types used in knoppy web production.

f	ρ_f (g cm^{-3})	l_f (mm)	D_f (µm)
Southdown wool	1.314	55 (cut)	29.7
PLA–51	1.210 - 1.430 [53]	51	21.6 - 19.9 (4 Denier)
PLA–32	1.210 - 1.430 [53]	32	21.6 - 19.9 (4 Denier)

Table 2.2: Number ratios for different fibre type blends.

f	f'	$m_f/m_{f'}$	$\rho_f/\rho_{f'}$	$l_f/l_{f'}$	$D_f/D_{f'}$	$N_f/N_{f'}$
		5.667 (85:15)				2.565 (72:28)
Southdown wool	PLA–51	4.000 (80:20)	1.086 - 0.919	1.078	1.375 - 1.492	1.811 (64:36)
		2.333 (70:30)				1.056 (51:49)
		5.667 (85:15)				1.610 (62:38)
Southdown wool	PLA–32	4.000 (80:20)	1.086 - 0.919	1.719	1.375 - 1.492	1.136 (53:47)
		2.333 (70:30)				0.663 (40:60)

reasonably well understood; it is possible to create different kinds of knops targeted at different products. There are two obvious controllable parameters:

- *The fibre blend*—the types and ratios of the wool and PLA fibres used have a significant effect on the softness and resilience of the knops.
- *The fibre length distribution*—one of the primary factors in the size of the knops. It also affects their formation; with even a small fraction of long fibres, the resulting knops are observed to form "tails," and be less well-formed. This enables knops to be incorporated into e.g. yarns, but is less desirable when creating knoppy web for bedding fill because the knops are less discretised.

There are also several other key controllable parameters that cannot be discussed here (as they are the proprietary know-how of the company).

Knops are very different to neps—the latter are entangled/felted fibrous clusters, while the former are spherical clusters of curled fibres. Suffice to say that while neps have been studied extensively (e.g. [54, 55]), there is no existing literature describing knops or the knop-forming process. A theoretical treatment of the knop formation process is outside the scope of this thesis, but the knops themselves will be modeled as part of the development of a knoppy web model in chapter 4.

2.7 Airlaying

The knoppy web itself is formed by taking the knop and web components, blending them together, and then feeding the blend into an airlay machine. The physics of blending was discussed in section 2.5, and differs little here; suffice to say that the primary concern is mixing the knops and web effectively without destroying the structure of the knops.

There exist a variety of airlaying technologies used for differing purposes [56], however only those that enable production of high-bulk products are of any significance to knoppy web production. Figure 2.18 shows the airlay that is currently used by FibreTech New Zealand Ltd to produce knoppy web. The general principle follows that of carding (section 2.3): the fibre blend to be airlayed is condensed in a hopper, and then a high-speed licker-in expands the blend into an airstream. The aerated blend flows through an outlet and against a pair of perforated rollers (fig. 2.19), allowing the air to escape and condensing the blend into a batt.

For pure fibrous blends, it is claimed that the DOA airlay provides an excellent random distribution of fibres throughout the batt, due to the air-blowing system used [56]. However, in the case of knoppy web blends, the licker-in has a significant effect on the structure of the end product. The licker-in is

Figure 2.18: A DOA airlay machine producing knoppy web.

Figure 2.19: Close-up of the outlet of the DOA airlay.

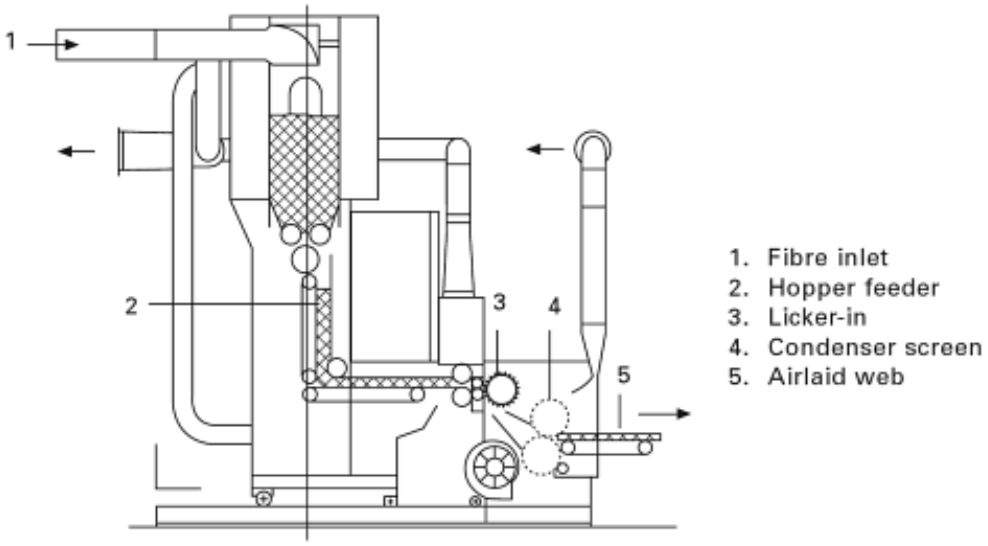


Figure 2.20: Schematic view of the DOA airlaying system [56].

situated above the airstream (fig. 2.20). Opened fibres are released from the licker-in by centrifugal force; the knops are significantly denser than the web fibre, and are flung further into the airstream. This means that the knop:web ratio increases across the thickness of the end product, with more knops at the bottom and more web at the top. This will affect the compressional properties of the product (because the stronger knops are mostly on one side), but also the handle and feel—the upper side of the knoppy web will have a softer feel and more uniform appearance.

This aberration can in fact be used to our advantage. By placing two sheets of knoppy web together with the knop-rich sides facing inwards, products can be created that have a softer handle (from the higher web fibre content at the surface), but with the structural benefit of the knops at its core. The thickness of the product would be increased, but this can be mitigated by laying the initial sheets at lower weights. Modern airlay machines can in fact be designed to perform this automatically, simply by having two licker-ins spinning in opposite directions, laying the fibre blend from above and below.

2.7.1 Target vs. actual weights

One of the important end properties of a knoppy web is its "weight," measured in grams per square metre. This is an important quantity in bedding and apparel products, as it affects many user-relevant properties such as warmth and comfort. Duvets are regularly marketed to users by their weight, usually with summer and winter weights. Thus it is important that knoppy web can be produced to a particular target weight, and the key control point for this is at the airlay.

In practice, however, target weights must be taken as a guideline only for the production line. It is not possible to reliably produce an exact weight specification, simply because of the complexity of airlay control. There are several different parameters that can be tuned on a DOA airlay:

- Hopper feeder speed.
- Licker-in speed.
- Airflow speed.
- Condenser screen height.
- Exit conveyor speed.

The exit conveyor speed has perhaps the simplest relationship: speeding up the conveyor lessens the density of the web. The condenser screen height puts a restriction on the potential thickness of the knoppy web. Beyond this, the relationships between the various parameters become increasingly non-trivial to specify precisely, particularly when factoring in the effect of any non-uniform blending of the knoppy web blend. The only reliable way to determine the actual weight produced is to run the airlay for a few metres and then measure the output—but the process of starting and stopping the airlay can itself affect the output weight. Achieving the target weight therefore comes down primarily to operator skill and experience.

2.8 Bonding

The final stage in the production of knoppy web is the bonding stage. The unbonded knoppy web is passed through a continuous bonding oven, which is set to a temperature that lies in between the melting points of the two components of the PLA fibres (section 1.2). The heat melts the outer component, causing the fibres to merge with other PLA fibres in contact with them. As the knoppy web exits the bonding oven, the temperature drops, forming PLA-PLA fibre bonds.

PLA fibres have lower friction at fibre contacts, and slip easily under load, so an unbonded knoppy web has much less cohesion than a regular web. This means that excess handling of the knoppy web prior

to bonding can seriously compromise its structural integrity. The ideal position for a bonding oven is directly after the airlay, so that the two stages can be performed sequentially. Barring that, the unbonded knoppy web would have to be rolled up and transported to the bonding oven. This would enable it to potentially sag under its own weight, pulling it apart and introducing holes that could compromise its thermal properties.

Bonding the PLA fibres turns many slipping contact points into non-slipping ones, creating a skeleton amongst which the wool fibres are threaded and interlocked. Given the relatively high number of PLA fibres in knoppy web (section 2.5.5), this bonded substructure is expected to be extensive. It is important that effective micro-scale blending (section 2.5) has been performed and the PLA fibres well-dispersed, both to ensure the pervasiveness of the skeleton and to avoid the generation of large hard lumps of PLA in the final product.

The two parameters available within the bonding process are the chosen bonding temperature, and the time spent in the oven. The melting point of the outer sheath of bicomponent PLA is about 130 °C, so it is necessary to raise the internal temperature of the unbonded knoppy web above this point. This necessitates a higher oven temperature and a longer in-oven time, particularly for bulky knoppy web due to the fact that the outer regions of the knoppy web will insulate the inner regions. At the same time, the temperature at the outer edge of the knoppy web cannot be raised too high, or the wool will potentially suffer discolouration [57].

2.9 Strategy and issues

As was alluded to in section 2.2, the sponsor company, FibreTech New Zealand Ltd, leverages all of the above tools and techniques for its knoppy web production strategies. The particular strategy enacted depends on the recipe and production volume, but the process outlined in fig. 2.1 is generally accurate.

The choice of card is an obvious control strategy, albeit one that is mostly decided by the particular fibre blend being used. The particular card used for each blend, and its particular settings (section 2.3), are outside the scope of this thesis; they have been determined through the decades of prior product development conducted at the factory.

Pre-melling is used to help blend the PLA fibre into the wool. The PLA fibre is passed through the opener with a roughly equal weight of the wool component, and the opened fibre is then carded. This results in a fibre pre-blend with a uniform wool:PLA ratio of 50:50 by weight, although the number of PLA fibres is significantly higher (section 2.5.5). In a subsequent step, the pre-blend is opened with the remainder of the wool component (which has separately been opened and carded) and then carded to obtain a final sliver with a wool:PLA ratio of the desired target. Taking a target wool:PLA ratio of 85:15 as an example, pre-melling turns a single 85:15 blending step into a 50:50 blending step followed by a 70:30 blending step.

The major sticking point with blending is achieving the desired target ratios. The process used for blending the PLA fibre into the wool fibre stock uses the opener and card three times each, with the primary ratio blending point occurring at the opener. Similarly, the process for blending knops with the web carrier fibre is to weigh out the desired ratio and pass the two components through the opener. There are several issues with this process:

- This is not a particularly efficient process; the time it takes to prepare the correct blend of components in the small feed hopper of the opener (section 2.5.2) is non-negligible.

- It assumes that the fine PLA fibres will blend more effectively into a smaller quantity of wool fibres. More specifically, if the pre-melling is done with an initial wool:PLA ratio of 50:50 by weight, there will be a larger number of PLA fibres present (section 2.5.5).

- Opening and carding all the fibre twice increases processing losses due to fibre breakage. This adds to the overall cost of the operation.
- Additionally, there is no guarantee that the sliver emerging from the first pre-melling step is actually at the target ratio. Not that there is any guarantee of this for any carding step, but the two-step measuring process increases the likelihood that the resulting web blend is not at the target specification.

Nevertheless, with the opener being the entry point to FibreTech New Zealand Ltd's production line, it is the only place where component ratios can really be set. Alternative blending tools (if there were any that didn't have these issues) would incur a large capital cost that is not economic given FibreTech New Zealand Ltd's scale of operations.

There is also the existing knowledge base to consider. All of FibreTech New Zealand Ltd's products, and the knowledge and IP built around them, have been developed using the opener as the initial component blending point. Even if the final product's component ratios do not match the recipe, its properties have been associated with the expected ratios. It is not outside the realm of possibility that altering the initial blending strategy could change the properties of the end product in unexpected ways. Ideally it would be for the better (by, for example, resulting in more even mixing of wool and PLA fibres), but that is not guaranteed.

Shortening the fibre stock is essential for knoppy web production, given that long fibres have been seen during prior product development to be detrimental to knop formation. It is a particularly important control strategy when the knops are desired to be small, due to the dependence on fibre length (section 2.6). Cutting is therefore utilised due to the ease of control over the cutting length (by simply adjusting the speed of the sliver feed relative to the cutting blade frequency), and the tendency towards a shorter fibre distribution than stretch-breaking (section 2.4). The prevalence of short fibres (section 2.4.1) is in some sense beneficial, although for particularly short cutting lengths it can be necessary to intentionally select a longer PLA fibre length, to ensure that there is still sufficient fibre length to have cohesive bonding (section 2.8).

Blending via cutting (section 2.5.4) is a useful strategy for controlling the fibre blend, and is used in blends where a weight ratio can be represented by a ratio of slivers. Implicit in this method, however, is the assumption that the linear densities of sliver A and sliver B are equal. In practice this assumption is reasonable at FibreTech New Zealand Ltd, for two reasons:

- The slivers that are to be cut and combined usually have about 50% of their fibre content in common, due to pre-melling (section 2.5.3).
- If the linear densities do differ (due to one sliver containing a particularly bulky fibre, or due to differences in carding parameters), this would result in there being excess sliver of either A or B. The collected cut fibre would therefore initially have a blend ratio that is light on A or B, but later switched to being solely comprised of that sliver. However, when subsequently fed into the opener's feed hopper (via bales or airstream) this would manifest as a layer at the top or bottom of the silo of pure A or B, which would be blended into the remainder of the fibres via the silo blending step (section 2.5.2).

Without an on-site bonding oven, there is little else that can be done but transport the unbonded knoppy web to an off-site bonding oven as carefully as is practical. However, the use of an in-line bonding oven is not without difficulties. One of the control parameters for the bonding stage is the time spent inside the oven (section 2.8), which is controlled by the speed of the oven's conveyor. With an in-line bonding oven, this conveyor must run at the same speed as the airlay's exit conveyor—which is a key control parameter for the knoppy web's weight (section 2.7.1). Further testing would be necessary on-site to ensure that the bonding process is adequate over the range of conveyor speeds employed on the airlay,

and whether the bonding oven temperature can be adjusted to compensate, or the airlay parameters need to be altered. Either way, significant care must be placed on operator training.

2.10 Summary

In this chapter we have examined the various processing stages in the production of knoppy web, and extracted the relevant physics underlying them. The available control strategies and their issues are well-understood, and the process for creating knoppy web has been shown to have a solid basis in the underlying physics.

Recalling our hypotheses in section 1.4, one of the key properties that we are interested in is bulk retention. It is clear that both blending and bonding play important roles in the bulk retention of knoppy web, and their physics is well-understood. However, much less is known about the physics of knops, and it is here that we focus our attention in the next two chapters.

Chapter 3

Evaluation of literature

3.1 Introduction

In this chapter, we set the groundwork for explaining why knoppy web performs as well as it does. We are most interested in the bulk retention properties of the knoppy web; as the previous chapter surmised, the bulkiness of the knoppy web is a product of the individual physical properties of its components—the web and the knops—as well as how they are blended together. There is no precedent in literature for the kind of model required to describe the physical behaviour of knoppy web under compression. A model of knoppy web must at its core describe both the knops and the web. We must therefore first understand how these components themselves behave.

Section 3.2 outlines the possibile avenues for modelling a knop. Section 3.3 then examines the historical precedent for compressional models of random fibre assemblies[1]; from this we select a model to describe the web component of knoppy web.

Key original contributions in this chapter:

- A fresh presentation and discussion of the assumptions within the van Wyk model of fibrous assemblies (section 3.3.1).
- An energy method representation of the van Wyk model (section 3.3.4).

3.2 Micromechanical models of knops

As stated in section 2.6, there is no existing literature describing the compression of spherical clusters of fibre, and therefore no basis for modelling knops. We must therefore construct our own model if we are to progress with a model of knoppy web.

There are several possible approaches for developing a model of the compression of a knop. One choice would be a fibre-based model, where the behaviour of the knop is elucidated from the behaviour of its consistent fibres in much the same way as has been done for general random fibre assemblies. The complexity of this approach however is vast, both theoretically and computationally. Models of random fibre assemblies (discussed in the next section) have the advantage of being able to assume that the dimensions of the assembly are much much larger than the diameter of the fibre; the models can therefore use techniques such as the orientation density function (e.g. [58]) to greatly simplify their analysis. This assumption is not valid for a knop, which has a typical diameter of less than 1 cm.

An alternative approach is to generalise the knop as a spherical object. A search for literature on the compression properties of spheres reveals two potentially relevant categories of study: hollow spherical shells, and solid spherical particles.

[1] The web component of the knoppy web product is effectively a random fibre assembly, and will be modeled as such.

3.2.1 Hollow spherical spheres

The compression of hollow spherical shells under an applied external force has been studied by several workers [59–61]. This work can be broadly split into two types, based on whether the considered displacement of the shell is small or large. For small displacements, the shell is considered to approximately maintain its spherical shape. This allows several simplifications to be made about the problem domain:

- There is no deformation in the middle of the plane of the shell—it remains neutral during bending.
- Points of the shell lying initially on a normal-to-the-middle plane of the shell remain on the normal-to-the-middle surface of the shell after bending.
- The normal stresses in the direction transverse to the shell surface can be disregarded.

The resulting configuration can generally be fully described in terms of the stresses within the surface and the forces along it. However, a "small displacement" is defined as the shell being displaced by no more than its thickness. Beyond this, it is necessarily deformed and the assumptions outlined above become invalid.

Deformations of spherical shells under compression have been studied within the context of both thin-walled shells such as "ping-pong" balls [60, 61] and thicker-walled objects such as tennis balls [62]. While deformation occurs initially as a flattening of the shell shape from spherical, the lowest energy configuration rapidly becomes a buckled state, with a first-order transition between the flattened and buckled configuration occuring at a deformation close to twice the thickness of the shell [60].

3.2.2 Solid spherical particles

As with hollow spherical shells, study of the compression of solid spherical particles can be split based on whether small or large displacements are considered. The axisymmetric small deformation of solid spherical particles has been of interest for some time. The first correct three-dimensional elasticity solution was developed by Sternberg and Rosenthal [63], who considered a sphere under a pair of equal and diametrically opposite concentrated loads. This was later generalised to accommodate any finite number of arbitrarily-oriented concentrated loads by Guerrero and Turteltaub [64].

A recent study of soft spherical particles [65] developed theoretical models for the large deformation of both incompressible particles (rubber) and compressible particles (sponge). Two models were proposed, which differed on whether the profile of the lateral surface (non-contacted surface of the particle) was assumed to be spherical or elliptical in shape. Comparisons to experimental data showed that while both models made acceptable predictions, the elliptical profile gave the best agreement.

3.2.3 Discussion

For a model to describe a knop successfully, it must incorporate several key points:

- Knops can undergo compression of the order of their entire diameter.
- Knops are not observed to buckle under compression.
- Knops are composed of curled fibres, and therefore do not have a uniform density.

The spherical shell models that assume small deflections are obviously unsuitable for simulating knops, because they exclude the first point. Those that allow large deflections incorporate buckling, which makes perfect sense for a continuous shell. But per the second point, knops do not show any obvious buckling. The "wall" of a knop is comprised of a random pseudo-two-dimensional layer of individual fibres. Even with the fraction of bonded PLA fibres forming a fixed scaffold within the knop, the fibres have a degree of transverse freedom that is not shared with transversely-isotropic membranes—both via fibre movement,

and the ability for fibres to undergo in-plane bending into voids within the knop. This means that their fibrous structure can take up in-plane membrane strain at a much lower energy cost than a real continuous membrane. Consequentially, these models are not suitable for knop simulation either.

The third point excludes the solid spherical particle models. Their premise is essentially that the particle mass between the compression surfaces can be treated like a uniform mass, and compression of the particle either results in a constant-volume shape change (for incompressible particles), or distributes strain (roughly) uniformly through the particle (for compressible particles). The latter of these is essentially the same idea behind models of fibre assemblies (discussed in the next section). But the internal structure of knops prevents us from assuming either constant-volume compression or assuming constant strain throughout the knop along the compression axis.

As none of the existing models are suitable, we must instead construct one from scratch. An ideal compromise would be to have a model that represented the knop as a spherical membrane, but with assumptions such that it does not buckle. This is the path that we will take in section 4.2.

3.3 Micromechanical models of random fibre assemblies

Why does a random assembly of fibres have bulk? The volume of the assembly is often an order of magnitude larger than what would be predicted from the mass of the fibres and their density. The reason, of course, is that the fibres are not perfectly packed. Random variations in the properties of each fibre (such as length and crimp), combined with their pseudo-random[2] distribution and orientation, means that the fibres instead are only touching neighbouring fibres at discrete positions. The regions of fibre between the contact points create pores within the fibre assembly [66].

Let us now apply a uniform pressure to the top of the assembly. The height of the assembly will decrease until the strain developed within it balances the external pressure. Intuitively, the strain will be distributed across all the fibres in the assembly, and will result either in fibres bending or moving within the assembly. The relationship between the pressure applied to the assembly, and the change in the volume of the assembly, can be measured experimentally; therefore, an ideal model will make a prediction about this relationship that can be tested.

There are two approaches that can be taken when analysing the micromechanical behaviour of a random fibre mass under compression. The first is the force method: forces on the individual fibres or fibre segments are considered, and by balancing all forces at a particular stage of compression, the behaviour of the assembly can be elucidated. This method can be used to paint an accurate picture of the internal strain within the fibre assembly. However, as the geometry of the system becomes more complex, the system becomes much harder to simulate.

The second approach is the energy method. The strains on the system are partitioned into orthogonal components or independent regions, and the energy for each component is calculated at different stages of compression to first order[3]. The separate energy terms can then be summed to obtain the total energy of the fibre assembly. If the various energy terms can be assumed to be elastic, then the minimum energy principle can be applied: a system can be in elastic equilibrium only when the total energy of the system is at a minimum. Therefore by minimizing the total energy, the actual behaviour of the fibre assembly is obtained. It is less trivial to compare models using this method to standard experimental pressure-volume curves, and the models also lose the finess of force method models. However, energy method models are more amenable to examining the "big picture"—how the assembly behaves as a whole. It also becomes

[2]The fibre orientation is generally not truly random, particularly for air-layed fibre assemblies.

[3]For the energies of the different components to be calculated correctly, it is important that they are independent, inasmuch as they should only depend on the current geometry. Thus the energy method is generally more applicable to small strains; at large strains, higher-order coupling between the components can occur that is not accounted for, diminishing the accuracy of the method.

trivial to apply simplifying assumptions; complex contributions to the total energy can be replaced or ignored for a simple model, and then re-integrated as desired.

There are many complicating factors to take into account. Only a small percentage of fibres will be perpendicular to the direction of compression, so different fibres will have different fractions of force applied. And because none of the fibres are stuck together, there will be a percentage of fibres that move or slip through the assembly instead of bending. This will have a variety of effects on nearby fibres. How the fibres move (whether by deflection or slippage) will also depend on many factors, such as the crimp of the wool, the properties of the fibres, and the overall randomness of the assembly.

In the following sections, we look at the historical development of micromechanical models of random fibre assemblies. We examine the various intellectual steps that have been taken. We carry out fresh analysis and evaluation of the models. Finally, we establish the theoretical base upon which our knoppy web model will be developed.

3.3.1 The van Wyk model

The first micromechanical model of a random fibre assembly was pioneered by van Wyk [67]. Originally intending to improve on earlier empirical work by M. and J. Eggert [68], van Wyk developed a new theory from the forces on the constituent fibres. He started with the following assumptions:

Assumption 1. *Only fibre bending was considered; slippage, twisting and extension were neglected.*

Assumption 2. *The fibre elements were assumed to be randomly distributed, randomly oriented and uniformly packed.*

Assumption 3. *Inter-fibre friction was neglected.*

By assuming away various complicating factors, van Wyk could focus on the core micromechanical behaviour—the bending of fibres. To calculate the overall compression of the fibre assembly under a given pressure, van Wyk simply calculated the behaviour of a single bending element, then integrated over all contact points in the assembly to get the total.

The van Wyk model has become the basis for nearly all subsequent research efforts, in part due to its composable nature. The model consists of three distinct parts:

1. Calculating the behaviour of a single bending element.
2. Determining the number of contact points in the assembly.
3. Calculating the behaviour of the assembly by modeling the continuum strain.

3.3.1.1 Deformation of the bending element

In any fibrous assembly, fibres will make contact with other fibres at various positions and angles. Imposing an axial direction to the assembly, we can observe that some contact points will be "above" the fibre they contact, and others will be "below" the fibre. In the face of downward pressure along the axis, the lower contact points act as support points, while the upper contact points will be imparting the force. Let us consider the two possible arrangements of a top contact point relative to the bottom contact points on a fibre:

1. The top contact is directly above a bottom contact point. In this case, the force on on the fibre from the top contact point will be transferred completely through to the bottom contact point, without any relative movement of the contact points.

2. The top contact point is not above any bottom contact points. In this case, the top contact point will either be between two bottom contact points, or between one bottom contact point and one end of the fibre (with the latter being statistically less common). In either case, the force on the fibre from the top contact point will cause it to bend.

Therefore, at the simplest level, the compression properties of a random fibre assembly can be described in terms of the bending of segments of fibre.

The original van Wyk bending element was developed by considering the fibre assembly as an ideal stack of weightless rods, all of uniform diameter D. Figure 3.1 shows a two-dimenional cross-section of the stack. The rods in each layer are equally spaced a distance $2b$ apart, and are perpendicular to the rods in the layer below. Every second layer is offset by a distance b. The stack of rods can be broken into a series of bending elements; one such element is shown in fig. 3.2. Each element is supported by two contact points a distance $2b$ apart, and has a contact point at the midpoint of the element. This construction implicitly requires the following assumptions:

Assumption 4. *The fibres can be represented by fibre elements with mean length b.*

Assumption 5. *The fibre elements are all initially straight.*

Assumption 6. *Contact points always alternate sides along a fibre, such that every bending element has just one contact point in its centre.*

This is a greatly simplified model of a real fibrous mass, where fibre contacts are not evenly spaced, and the forces distributed onto the bending elements will not necessarily be perpendicular to the fibre.

Under compression, a force F will act on this contact point, deflecting the beam by a distance y. Another assumption is made:

Assumption 7. *The fibre elements are treated as (initially) straight beams with built-in ends.*

Under this assumption, the relationship between the force F and deflection y is given by the standard equation for bending a cylindrical rod [67],

$$F = \frac{24IE_f}{s^3}y, \tag{3.1}$$

where

$$I = \frac{\pi D^4}{64} \tag{3.2}$$

is the moment of inertia of cross-section of the fibre, E_f is the Young's modulus of elasticity of the fibre, and s is the length of the fibre element. D is the fibre diameter, per fig. 3.1.

This standard equation has two issues:

- The fibre element length must be averaged, which will rescale the equation.
- It requires that the force is applied perpendicular to the fibre element. By van Wyk assumption 2 (and of course in general), this will not be the case.

To circumvent these issues, van Wyk made the following assumption:

Assumption 8. *The deviation of a general fibre assembly from the ideal case of all fibres being perpendicular to the applied force can be accounted for by a statistical constant of proportionality.*

Therefore we replace s with b, and replace the numerical constant with an unknown factor k; thus

$$\mathrm{d}F = \frac{kIE_f}{b^3}\,\mathrm{d}y, \tag{3.3}$$

where we have written the equation in terms of incremental force evolution.

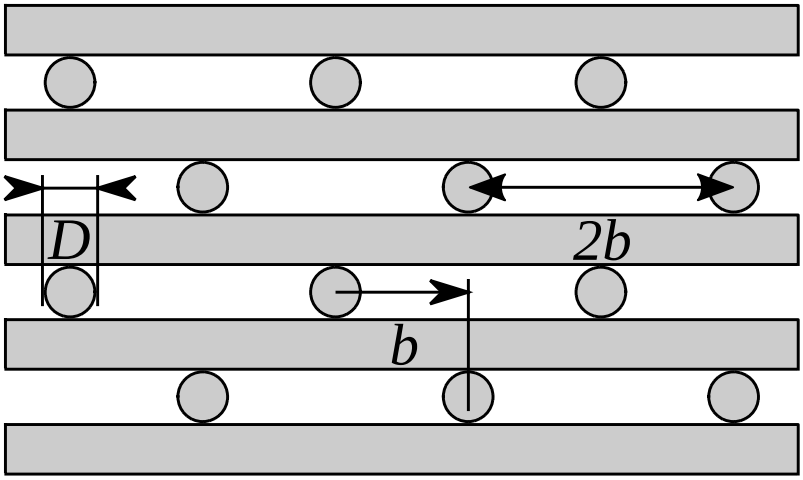


Figure 3.1: A van Wyk pile of rods.

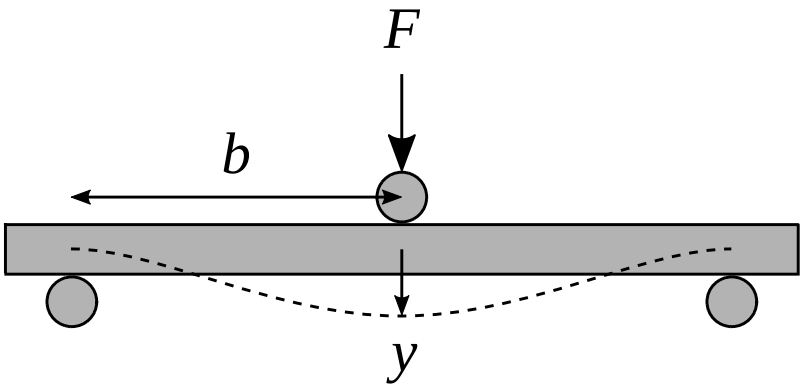


Figure 3.2: The van Wyk bending element.

3.3.1.2 Counting the number of contact points

How does one count an uncountable number? As with the theoretical treatment of carding in section 2.3.1, the answer is to look at the system stochastically. A contact is formed when two fibres touch; thus the number of contact points can be determined by first establishing the probability of two fibres touching in the assembly.

The method that van Wyk used takes inspiration from the classical theory of kinetic gases, and its concept of "mean free path," or the average distance between collisions. Let there be an assembly of fibres with diameter D placed randomly in a cylinder of unit cross-sectional area and height v (which therefore is also the volume of the cylinder). Let there also be a particle of diameter D travelling vertically through the cylinder. The particle will collide with a fibre when their centres are within a distance D of each other, giving the particle an "effective collision area" of a circle with diameter $2D$. As the particle travels a distance v through the cylinder, it sweeps out a volume $\pi D^2 v$, and any fibre within that volume will collide with the particle.

The number of fibres in the collision volume can be estimated by the ratio of the area of fibre presented to the particle within that volume to the cross-sectional area of the volume. By assuming that the fibres are randomly distributed, the ratio is equal to the total area of fibre presented to the particle divided by the unit volume of the cylinder. If the fibres are also assumed to be randomly oriented, the presented area of a fibre is the effective area projected onto a plane perpendicular to the direction of motion of the particle. If θ is the angle between any one fibre element and the direction of motion of the particle, then the effective area of the element is equal to the actual area multiplied by $\sin\theta$. Generalising this to the entire fibre assembly is as simple as taking the spatial average of $\sin\theta$, which is $\pi/4$. Recalling that the collision diameter of a fibre is $2D$, the total area of fibre is $2Dl$, where l is the total length of fibre in the assembly. Therefore, the effective area presented to the particle is

$$2Dl\frac{\pi}{4} = \frac{\pi Dl}{2}. \tag{3.4}$$

Dividing this by the unit area of the cylinder, this is therefore also the number of fibres in the collision volume. From this, it can be established that the mean free path of the particle travelling a distance v is

$$\ell = \frac{2v}{\pi Dl}. \tag{3.5}$$

Let $L = l/v$ be the length of fibre per unit volume of assembly, and make the following assumption:

Assumption 9. *The trajectory of a fibre in the assembly can be represented by the path of a particle moving through the assembly, and the contact points along that fibre correspond to collisions of the particle with fibres.*

Observing that that the mean free path of the particle travelling through the assembly is equal to the average distance between fibres along the path, the fibre will be split up by these contact points into a series of elements, and the mean element length b is

$$b = \frac{2}{\pi DL}. \tag{3.6}$$

By van Wyk assumption 4, the number of contact points per unit volume of the assembly is simply

$$\begin{aligned} N &= \frac{1}{2}\frac{L}{b} \\ &= \frac{\pi DL^2}{4}. \end{aligned} \tag{3.7}$$

The factor of half comes from the observation that each contact will map to two locations on the total fibre length.

3.3.1.3 Modelling the continuum strain

The final part of the model involves scaling the individual bending elements up to the full assembly. Recall the cylinder of unit cross section from the previous section. Van Wyk makes the following two assumptions:

Assumption 10. *The fibrous assembly can be divided into layers of thickness c, where c is the vertical extent of a fibre element.*

Assumption 11. *The continuum strain is equal to the differential strain on an infintesimal volume.*

By van Wyk assumptions 2 and 10, the fibre elements are equally distributed amongst the layers. Therefore, since the total number of fibre elements in the assembly is l/b, the number of fibre elements in a single layer is

$$n_c = \frac{c}{v}\frac{l}{b}. \tag{3.8}$$

By van Wyk assumption 11, the incremental change in volume of the whole assembly is

$$dv = -\frac{v}{c}dy. \tag{3.9}$$

It should be noted that this limits the applicability of the model. By assuming a unit area cross-section and defining the volume change purely in terms of the axial compression, the model cannot account for lateral spreading of the random fibre assembly, ie. the Poisson's ratio. Thus, the model is only applicable either when there is a fixed immovable transverse boundary, or when the assembly can be assumed to have a periodic or infinite boundary.

3.3.1.4 The van Wyk equation

The above three parts can be combined to arrive at the van Wyk equation that describes the volume change of a random mass of fibres under an applied external pressure.

Pressure is force per unit area. By van Wyk assumption 11, the pressure is constant througout the fibre assembly, and can be calculated from the behaviour of a single layer. Recalling that the previous sections assumed unit cross-sectional area, an incremental change in the applied pressure is simply the product of the number of fibre elements in a layer and the incremental force applied to each element. Combining this with eqs. (3.3), (3.8) and (3.9),

$$\begin{aligned} \mathrm{d}p &= n_c\,\mathrm{d}F \\ &= \frac{c}{v}\frac{l}{b}\,\mathrm{d}F \\ &= \frac{kIE_f cl}{vb^4}\,\mathrm{d}y \\ &= -\frac{kIE_f c^2 l}{v^2 b^4}\,\mathrm{d}v. \end{aligned} \tag{3.10}$$

By van Wyk assumption 2, the mean value of c^2 can be replaced with $b^2/3$ [67]. Combining this with eqs. (3.2) and (3.6),

$$\begin{aligned} \mathrm{d}p &= -\frac{kIE_f l}{3v^2 b^2}\,\mathrm{d}v \\ &= -\frac{kE_f}{3v^2}\left[\frac{Il}{b^2}\right]\mathrm{d}v \\ &= -\frac{kE_f}{3v^2}\left[\frac{\pi D^4}{64}\left(\frac{\pi l D}{2v}\right)^2 l\right]\mathrm{d}v \\ &= -\frac{kE_f}{3v^2}\left[\left(\frac{\pi D^2 l}{4}\right)^3 \frac{1}{4v^2}\right]\mathrm{d}v \\ &= -\frac{kE_f m^3}{12\rho_f^3}\frac{1}{v^4}\,\mathrm{d}v, \end{aligned} \tag{3.11}$$

where m is the mass of the fibre assembly and ρ_f is the fibre density (about 1.3 g/cm^3 for wool).

Integrating,

$$p = \frac{KE_f m^3}{\rho_f^3}\left(\frac{1}{v^3} - \frac{1}{v_0^3}\right), \tag{3.12}$$

where the numerical constants have been bundled along with "variations in element length, diameter, contour, elasticity, and other fibre characteristics" [67] into an overall constant K.

3.3.1.5 Assumptions and issues

In developing the model above, van Wyk made a large number of assumptions to simplify the derivation. Several of these assumptions are clearly unrealistic, as was recognised by both van Wyk and subsequent workers.

A 1990 review by Lee, Carnaby, Carr and Moss [69] examined the effect of the length distribution on the value of k. Under van Wyk assumption 4, the value of k is at most 6 times larger than for a random distribution of element lengths. That is, van Wyk's model overestimates the stiffness of the fibrous assembly. This was also reflected in their analysis of the distribution of the number of loading points between supports. A pile of rods with randomised arrangement (but equally spaced) has a lower stiffness than van Wyk's pile, but greater than one quarter of the van Wyk stiffness.

One of the main limitations of the van Wyk model is the accuracy of the contact point count. Van Wyk assumption 2 states that the fibre assembly is randomly oriented; in a general fibre assembly, the fibres are not oriented randomly, which invalidates the determination of eq. (3.4). The inconsistency amongst van Wyk assumptions 2 and 8 is particularly jarring. It is physically inconceivable for all fibre elements to be simultaneously randomly oriented and perpendicular to the axis of compression.

Inherent in van Wyk assumptions 1, 3 and 4 is the requirement that contact points remain fixed—the fibres are treated as if they are glued together. This is an obvious simplification, and leads to models predicting force-displacement curves that are much stiffer than experimental curves. In reality, under increasing strain some fibre contact points will slip, causing the neigbouring bending elements to relax and change length.

Finally, van Wyk assumption 9 has particular significance. Recall that van Wyk derived the number of contact points by considering the trajectory of a particle through the fibre assembly. The crucial simplification is that when a particle collides with another particle, it will change its trajectory. The same does not apply for a fibre, and this difference illuminates the true issue with this method: it does not account for the steric hindrance of fibres.

As echoed by van Wyk assumption 8, it was claimed that these assumptions would require only simple corrections. In the next section, this claim will be evaluated on the basis of subsequent work.

3.3.2 Improvements on the van Wyk model

3.3.2.1 The orientation density distribution function

Komori and Makishima [58] provided the first major theoretical extension of the van Wyk model. Their work addresses the fact that general fibre assemblies are not randomly oriented, and aims to remove the model's dependency on van Wyk assumption 2. They consider straight fibres, but their work can equally be applied to straight fibre segments. Fibre orientations are specified in spherical polar coordinates, where θ is the polar angle, and ϕ is the azimuthal angle.

Komori and Makishima introduced the orientation density function, $\Omega(\theta, \phi)\sin\theta$. The function is defined such that the probability of a fibre lying within the infintesimal range θ to $\theta + d\theta$ and ϕ to $\phi + d\phi$ is

$$\Omega(\theta, \phi)\sin\theta d\theta d\phi. \tag{3.13}$$

From the definition of probability density functions, it follows that

$$\int_0^\pi d\theta \int_0^\pi d\phi \Omega(\theta,\phi)\sin\theta = 1. \tag{3.14}$$

Komori and Makishima then examined the contact point between a pair of fibres of length λ and diameter D. As with van Wyk's analysis, they consider that two fibres will make contact when their centres are within a distance D. Figure 3.3 shows a situation where a fibre with orientation (θ,ϕ) makes contact with another fibre with orientation (θ',ϕ'). The probability of such a contact forming is

$$p = 2D\lambda^2 \sin\chi, \tag{3.15}$$

where χ is the angle between the fibres, given by

$$\cos\chi = \cos\theta\cos\theta' + \sin\theta\sin\theta'\cos(\phi-\phi'). \tag{3.16}$$

The average number of contacts on an arbitrary fibre is therefore

$$n_v = \frac{2DN\lambda^2}{v} I, \tag{3.17}$$

where N is the number of fibres in volume v,

$$I = \int_0^\pi d\theta \int_0^\pi d\phi J(\theta,\phi)\Omega(\theta,\phi)\sin\theta \tag{3.18}$$

and

$$J = \int_0^\pi d\theta' \int_0^\pi d\phi' \Omega(\theta',\phi')\sin\theta' \left[1-\cos\theta\cos\theta' + \sin\theta\sin\theta'\cos(\phi-\phi')^2\right]^{1/2}. \tag{3.19}$$

Consequentially, they established that the mean free fibre length is

$$b = \frac{1}{2DLI}, \tag{3.20}$$

and the number of contacts per unit volume is

$$N = DL^2 I. \tag{3.21}$$

Komori and Makishima proved the generality of their model by showing that under the van Wyk assumptions, eq. (3.18) reduces to $\pi/4$, the spatial average used by van Wyk. In this case, eqs. (3.20) and (3.21) reduce to eqs. (3.6) and (3.7) respectively.

3.3.2.2 Randomly-oriented bending elements

Lee and Lee [70] leveraged the work of Komori and Makishima (in the previous section) to consider a randomly-oriented bending element, and remove the need for van Wyk assumption 8 to cover over the fact that force is not applied perpendicular to most bending elements.

Figure 3.4 shows the randomly-oriented bending element, with direction defined in the spherical polar coordinate system used by Komori and Makishima. As with the van Wyk model, there are three contact points equally spaced along a bending element of length $2b$; two of these are supporting the bending element, while the third applies a load to the centre of the bending element. The deflection of the bending element is also modeled as a straight rod. However, Lee and Lee chose to use the equation for a beam with free ends.

The difference comes in calculating the z-component of deflection. The vertical load on the ith bending element, C_i, can be split into a normal component C_{in} and a parallel component C_{ip}. As in the van Wyk model, extension of fibre elements (and thus the contribution of C_{ip}) is ignored. The normal component deflects the bending element according to the standard equation; this deflection δ_i can then be split into its Cartesian components δ_{ij}.

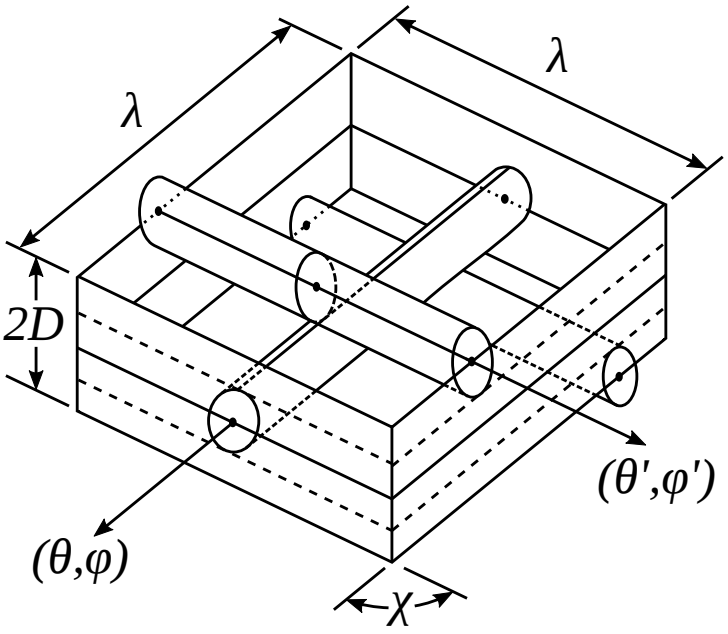


Figure 3.3: Contact of fibre B of orientation (θ', ϕ') with A of (θ, ϕ) and sweepings of the former on both sides of the latter, keeping the contact point and direction of the former unchanged [58].

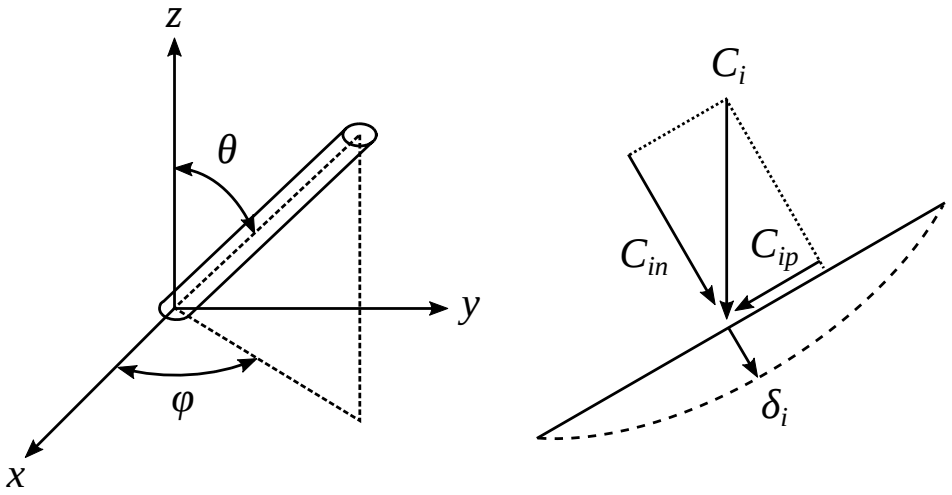


Figure 3.4: Direction of fibre, and deformation of a fibre element [70].

Keeping van Wyk assumption 11, the behaviour of the entire assembly can be determined by calculating $\bar{\delta_{ij}}$, the mean deflection along each axis. The fibres in the assembly are oriented according to the orientation density function. Lee and Lee derived the mean deflection along each axis as

$$\bar{\delta_{ij}} = \pm \frac{2C_i b^3}{3B} M_{ij}, \tag{3.22}$$

where B is the bending modulus of the fibre, and [70]

$$M_{11} = \int_0^{\pi/2} d\phi \int_0^{\pi/2} d\theta (1 - \sin^2\theta \cos^2\phi)\Omega(\theta,\phi)\sin\theta \tag{3.23}$$

$$M_{22} = \int_0^{\pi/2} d\phi \int_0^{\pi/2} d\theta (1 - \sin^2\theta \sin^2\phi)\Omega(\theta,\phi)\sin\theta \tag{3.24}$$

$$M_{33} = \int_0^{\pi/2} d\phi \int_0^{\pi/2} d\theta \sin^2\theta \Omega(\theta,\phi)\sin\theta \tag{3.25}$$

$$M_{12} = M_{21} = \int_0^{\pi/2} d\phi \int_0^{\pi/2} d\theta \sin^2\theta \sin\phi\cos\phi\Omega(\theta,\phi)\sin\theta \tag{3.26}$$

$$M_{23} = M_{32} = \int_0^{\pi/2} d\phi \int_0^{\pi/2} d\theta \sin\theta\cos\theta\sin\phi\Omega(\theta,\phi)\sin\theta \tag{3.27}$$

$$M_{31} = M_{13} = \int_0^{\pi/2} d\phi \int_0^{\pi/2} d\theta \sin\theta\cos\theta\cos\phi\Omega(\theta,\phi)\sin\theta. \tag{3.28}$$

The sign of eq. (3.22) is positive when $i \neq j$ and positive when $i = j$.

3.3.2.3 Slippage

We now turn to van Wyk assumptions 1, 3 and 4. Carnaby and Pan [14] took a crucial theoretical step towards resolving the issue of slippage in a fibre assembly under compression, by examining the forces present at a contact point.

Figure 3.5 gives a diagram of the contact point between two fibre elements under an applied external force, showing the forces present. The force is distributed over the assembly, with the jth contact point being subjected to a force C_j; this can be split into the normal force C_{jn} and the force parallel to the fibre C_{jp}. In previous models [70], C_{jn} is the force that deflects the fibre element underneath the contact point, and C_{jp} is ignored because all contacts are non-slipping. Carnaby and Pan reason that the jth contact will be non-slipping only while C_{jp} is less than the static friction force at that contact point.

This static friction force can be detected indirectly by measuring the force required to withdraw a single fibre from the fibre assembly at a constant rate. Recall from section 2.3.2 that F_{W0} is the value of the withdrawal force F_W per unit length at zero external pressure. A fibre with n contact points will be split into n fibre elements on average[4], and therefore the mean static friction force at each contact will be $F_{W0}b$ (recalling that b is the mean distance between contacts). Thus, the condition for a contact point to slip is when

$$C_{jp} \geq \mu C_{jn} + F_{W0}b, \tag{3.29}$$

where μ is the coefficient of friction between two fibres. Writing this in terms of the uniaxial force C_j on the contact point and the orientation of the bottom fibre,

$$C_j \cos\theta \geq \mu C_j \sin\theta + F_{W0}b. \tag{3.30}$$

It is clear (and intuitive) that there is a critical polar angle θ_c, and fibres with polar angles smaller than θ_c will be slipping contact points. Solving this for the limiting case $\theta = \theta_c$,

$$\sin\theta_c = \frac{-\frac{\mu F_{W0}b}{C_j} \pm \left[\frac{\mu^2 F_{W0}^2 b^2}{C_j^2} - (1+\mu^2)\left(\frac{F_{W0}^2 b^2}{C_j^2} - 1\right)\right]}{1+\mu^2}. \tag{3.31}$$

[4]The exact number for each fibre will be between $n-1$ and $n+1$ fibre elements, depending on where the end contacts are relative to the ends of the fibre.

Knowing θ_c, it is a trivial step to count the number of contact points that will be slipping or non-slipping under a particular uniaxial strain, simply by adjusting the integration limits in eqs. (3.18) and (3.19).

The above model is suitable for modeling the compression of a random fibre assembly, but does not accurately describe the recovery phase. From a modeling perspective, the simplest approach is to assume that the recovery phase is the reverse operation to the compression phase. This is an assumption common to all previous models, but by design it cannot be applied here. The release of bending strain in the fibre elements drives the recovery, and as Carnaby and Pan noted, the force applied by the fibre element (the lower fibre in fig. 3.5) to the contact point is a normal force. For the slipping contact points, this means that within the current model constraints there is no direct cause for reversal of the slippage, even when the external force is removed completely.

A potential method to model the recovery might then be to treat the assembly as a "classical" model with all contact points non-slipping, and only count those contact points that were non-slipping at the end of the compression phase. In reality, the recovery of surrounding bending elements will cause some fraction of slipping contact points to also recover, and so this method will predict a larger hysteresis than would actually be observed. Carnaby and Pan handled this by assuming that the contact points will slip "backwards" if the static friction force is exceeded by a finite value of C_{ij} directed up the fibre element. That is,

$$C_j \cos\theta = F_{W0} b.$$

The remainder of the recovery phase analysis continues as above.

3.3.2.4 Steric hindrance

Van Wyk assumption 9 is the final significant assumption that needs to be addressed—specifically, the fact that the model does not account for the steric hindrance of fibres. Komori and Makishima's development of the orientation density function similarly ignores steric hindrance when determining the probability of two fibres making contact (eq. (3.15)).

Lee and Carnaby [71] developed a new micromechanical model for uniaxial compression of a fibre assembly using the energy method (described in detail in appendix A). To model the fibre element length distribution, they used the gamma function [72]

$$f(l) = \frac{[(n+1)/\bar{l}]^{n+1}}{n!} l^n e^{-[(n+1)/\bar{l}]l}. \tag{3.32}$$

They define the value of n to be equivalent to the level of hindrance; increasing n shifts the length distribution towards longer element lengths, implying that the contact points on a fibre are on average further apart. This provides the model user with a parameter that can be fitted to experimental distributions, but hand-waves over the actual micromechanics. In particular, the number of contact points is not linked to this function, and is still overestimated.

Pan [73], and subsequently Komori and Itoh [74], developed a modified theory of fibre contact that accounted for steric hindrance:

$$p(o, o') = h(o, o') p_0(o, o'),$$

where $p(o, o')$ is the probability that two fibres with orientations o and o' will contact within the modified fibre assembly, $p_0(o, o')$ is the same probability for the ideal fibre assembly, and $h(o, o')$ is the hindrance factor. This factor was shown to be influenced by two concepts, formally defined by Komori and Itoh: the forbidden length and the forbidden volume.

The forbidden length was first introduced by Pan [73], and refers to the length of fibre in an assembly on which a contact cannot be made. Consider a new fibre being introduced into an existing fibre assembly. It is not possible for the new fibre to contact existing fibres where there is already a contact (length b_b in fig. 3.6a), so the length of fibre that is available will be less than the total length of fibre in the assembly. Pan argued that this would decrease the total number of contact points in the assembly, leading to a

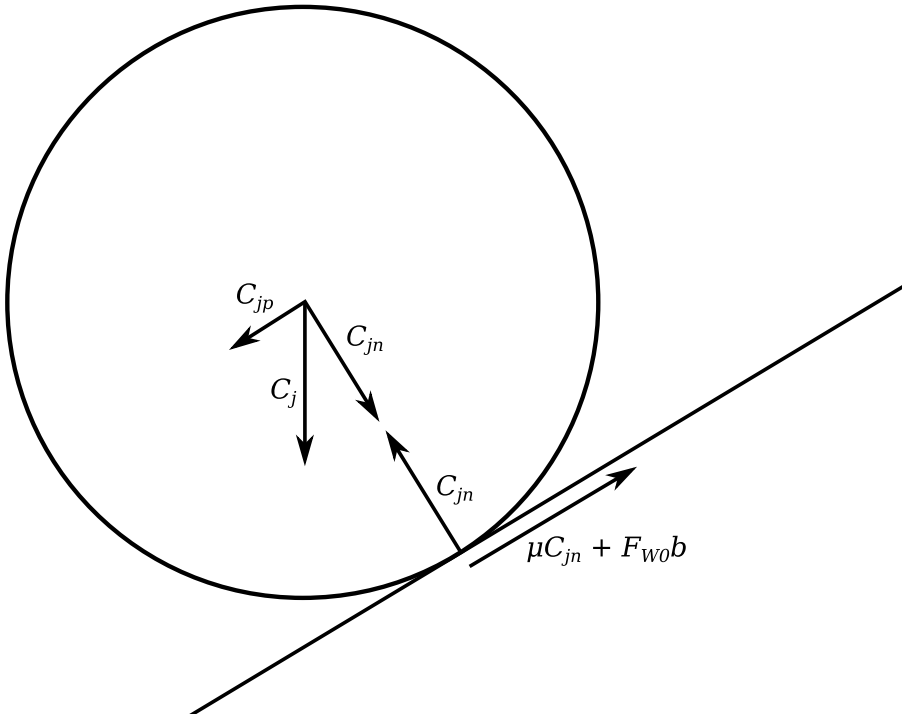
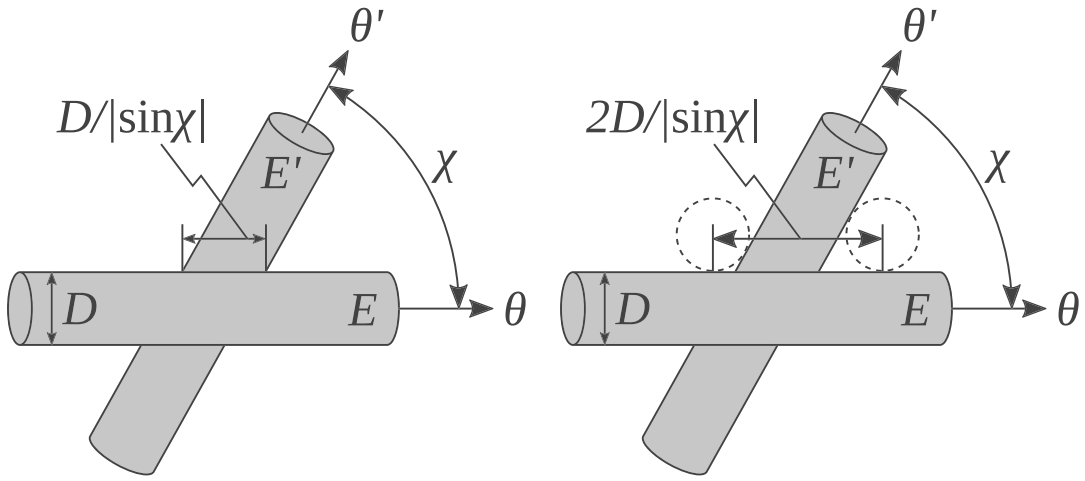


Figure 3.5: Diagram of a slipping contact point [14].



(a) As specified by Pan [73].

(b) As specified by Komori and Itoh [74].

Figure 3.6: The forbidden length.

longer mean fibre element length and the overall assembly being less stiff than prediced by Komori and Makishima's ideal theory. However, Pan understated the forbidden length by a factor of two; he defined the forbidden length as the fraction of the total fibre length occupied by contacts, whereas Komori and Itoh used the more valid definition that it is the fraction of the total fibre length along which a new fibre would intersect an existing fibre (fig. 3.6b).

The forbidden length acts as an inhibiting factor on the number of fibre contacts in a fibre assembly. However, as Komori and Itoh explained [74], there is another factor that comes into play. The forbidden volume is the volume fraction of the fibre assembly that a fibre element cannot occupy because, if it were placed there, it would contact some region of the forbidden length on the existing fibre assembly. Komori and Itoh surmised that this would act as a compensating factor against the forbidden length, simply because a fibre element randomly introduced into an allowed volume will be more likely to contact the fibres surrounding that volume when the volume is smaller.

Komori and Itoh showed that the hindrance factor could be recursively defined as

$$h(o,o') = \frac{1}{2}\left[\frac{1-2qv_F F(o)}{1-q^2\alpha v_F^2 G(o')} + \frac{1-2qv_F F(o')}{1-q^2\alpha v_F^2 G(o)}\right], \tag{3.33}$$

where

$$v_F \equiv \frac{N\pi D^2\lambda}{4v} \tag{3.34}$$

is the fibre volume fraction of the mass, $\alpha \equiv \lambda/D$ is the basic aspect ratio, $q \equiv 8/\pi$,

$$F(o) \equiv \int h(o,o')\Omega(o')d\omega',$$

$$G(o) \equiv \int F(o')\sigma(o,o')\Omega(o')d\omega'$$

and

$$\sigma(o,o') \equiv |\sin\chi(o,o')|.$$

3.3.2.5 Accuracy of the van Wyk constant

Van Wyk predicted that the key simplifications of his model could be handled by constants of proportionality. In most cases, he has been proved right:

- Accounting for non-random orientation of fibres added a simple multiplicative factor to eq. (3.7) that only depends on the orientation density function.
- The modifications required to adjust the deflection for non-perpendicular force (eq. (3.22)) are similarly multiplicative.
- Steric hindrance can be modeled by a non-trivial but independent hindrance factor.

Even the inclusion of slippage acts roughly as a dampening factor on the number of bending elements, at least during compression. The recovery phase is necessarily distinct, as hysteresis could not be adequately accounted for by a single constant of proportionality.

3.3.3 Discussion

The last four decades in particular have clearly been very fruitful, and the discipline continues to evolve [75]. But when viewed in the context of knoppy web, many of the extensions amount to unnecessary complications, and can be ignored for a variety of reasons.

Firstly, we must consider the issue of matching complexity. A model describing the behaviour of knoppy web under compression must be self-consistent. When choosing a particular model to use in the system, the level of detail it entails should be similar to other components, and they should have

common assumptions. As much as accuracy is important in a model, more important is that the model is reasonable; too much detail can be as detrimental as too little. In particular, any assumptions that are used need to be physically justifiable.

As surmised in section 3.2.3, a new model must be developed to describe a knop, and the model will contain various simplifying assumptions. These assumptions will by extension form part of the overall knoppy web model. We believe that the web model should match the knop model in terms of its complexity and assumptions, partly because it is more realistic for an assumption to apply to the whole knoppy web than just to the knops, and partly because it keeps the computational requirements of the model at an overall similar level.

There is some flexibility in the complexity of the web model. As stated above, the extensions to the van Wyk model were successfully handled with multiplicative factors. While these factors introduce additional non-trivial computation requirements, they are intrinsically compatible with the majority of existing web models. Thus, we can choose to work with a model that does not account for these factors, and they can be introduced later if and when they become necessary.

As much as the choice of web model is influenced by the knop model, it is also influenced by the other point of difference in knoppy web: the presence of PLA fibres. Two of our four hypotheses are about the effect of PLA on the knoppy web, and the PLA fibre can be found within both the knops and the web. We cannot ignore the PLA fibres within the chosen web model, and in fact we hypothesise that their presence actually makes simulation simpler.

Regular wool fibre assemblies exhibit large hysteresis in their compression-recovery curves, due to non-recoverable slippage at contact points. This was modeled by Carnaby and Pan, as described in section 3.3.2.3. The introduction of PLA fibre into the assembly only exacerbates the hysteresis, because PLA is a smoother fibre than wool and has less friction at contact points. As outlined in section 2.5.5, a fibre assembly with 15% PLA by weight could in fact have as much as 40% PLA by number. Therefore, an unbonded wool-PLA fibre assembly will undergo much higher levels of slippage during compression; combined with the fact that the PLA fibres store much less bending energy than wool fibres (being thinner), it is clear that the assembly will recover less and have a larger hysteresis.

However, when we pass the wool-PLA fibre assembly through a bonding oven (as happens with knoppy web), the situation is inverted. SEM studies presented in section 5.4.3.3 show that the PLA fibres readily bond to each other, but poorly with wool. We can therefore assume that all PLA-PLA contact points will be bonded, while the PLA-wool contact points can be treated in the same way as the wool-wool contact points. Under this assumption, the physical situation is drastically simplified.

Consider the case of a contact point in a pure wool assembly that has begun to slip. In Carnaby and Pan's treatment (section 3.3.2.3), a contact point that becomes slipping will continue to slip for all subsequent compression. This treatment assumes that the top fibre is physically able to continue slipping, which in a pure wool fibre assembly is a reasonable assumption, because the slipping fibre can deflect any other fibres it comes into contact with. Like all previous models, any new contact points formed during compression are ignored, and so this model remains self-consistent.

Now consider the case of a contact point in a bonded wool-PLA fibre assembly that has begun to slip. We can assume that any energy contributions from wool fibres that the slipping fibre comes into contact with can be ignored, as before. However, the slipping fibre can also come into contact with PLA fibres as it slips. On average, a fibre will slip some small distance before contacting a PLA fibre. However unlike before, the slipping fibre will not slip beyond the location of the PLA fibre. This is because if the slipping fibre were to slip further, it would cause the PLA fibre to go into tension—as assumed earlier, all PLA-PLA contacts are bonded, so each PLA fibre segment is effectively fixed. Thus after a short slipping distance, the slipping contact becomes a non-slipping contact again, and so the potential hysteresis can be considered to be negligible. Therefore we propose to ignore hysteresis, at least in this early knoppy web model.

Based on the above considerations, we have elected to use the van Wyk model to describe the web component of knoppy web. We will design the knoppy web model to be as modular as possible, so that the van Wyk model can be replaced in later extensions; however, we feel that it is sufficient for our initial purposes.

3.3.4 van Wyk energy method

Section 3.3 mentioned two approaches that can be taken to analyse the micromechanical behaviour of a system—the force method, and the energy method. The latter has a significant advantage in terms of composability: the system can be split into a series of geometrically-linked components, and energy terms calculated for each component separately. To ease the combination of the knop and web models, we elect to use the energy method, and therefore must extend the van Wyk model to calculate the energy of the web.

Recall from section 3.3.1.4 that van Wyk derived eq. (3.12) to describe the volume change of a random mass of fibres under an applied external pressure. Restated here,

$$p = \frac{KE_f m^3}{\rho_f^3}\left(\frac{1}{v^3} - \frac{1}{v_0^3}\right),$$

where K is the general constant, E_f is Young's modulus of elasticity of the fibre, m is the mass of the fibre assembly and ρ_f is the fibre density. E_f and ρ_f are standard parameters, and

$$m = \rho_w v_0, \tag{3.35}$$

where ρ_w is the packing density of the fibre assembly.

From classical mechanics we know that pressure is a measure of energy density—that is, $\mathrm{d}U = -p\,\mathrm{d}v$. We can therefore integrate from initial volume v_0 to compressed volume v_c in order to obtain the internal energy of the web:

$$\begin{aligned} U_{\mathrm{vW}} &= -\frac{KE_f m^3}{\rho_f^3}\int_{v_0}^{v_c}\left(\frac{1}{v^3} - \frac{1}{v_0^3}\right)\mathrm{d}v \\ &= \frac{KE_f m^3}{\rho_f^3}\left[\frac{1}{2v^2} + \frac{v}{v_0^3}\right]_{v_0}^{v_c} \\ &= \frac{KE_f \rho_w^3 v_0^3}{\rho_f^3}\left[\frac{1}{2v_c^2} + \frac{v_c}{v_0^3} - \frac{1}{2v_0^2} - \frac{v_0}{v_0^3}\right] \\ &= KE_f\left(\frac{\rho_w}{\rho_f}\right)^3\left[\frac{v_0^3}{2v_c^2} + v_c - \frac{3v_0}{2}\right]. \end{aligned} \tag{3.36}$$

Figure 3.7 plots the van Wyk energy against the relative change in volume (v_c/v_0), for some standard parameters (defined later in table 4.2). The energy behaves as we intuitively expect it to, and has a minimum of zero at $v_c = v_0$. For $v_c < v_0$ the inverse quadratic term dominates, and the strain energy increases asymptotically, consistent with the pressure-volume behaviour. Obviously the model makes no physical sense for the case $v_c > v_0$, as this corresponds to a negative pressure, which is not possible. Applying axial tension to a basic random fibre assembly will result in it being pulled apart. The fact that the van Wyk energy model gives positive strain energies for $v_c > v_0$ is a consequence of two assumptions: the continuum strain assumption (assumption 10 from section 3.3.1.5), and the assumption of unit cross-sectional area for the calculations. This creates an artificial strain for $v_c > v_0$.

3.4 Summary

The original model of random fibre assemblies developed by van Wyk [67] has been selected for modelling the web component of the knoppy web model. However, no existing model is suitable for describing knops. Instead, we will develop a model that represents the knop as a spherical membrane, but with assumptions such that it does not buckle.

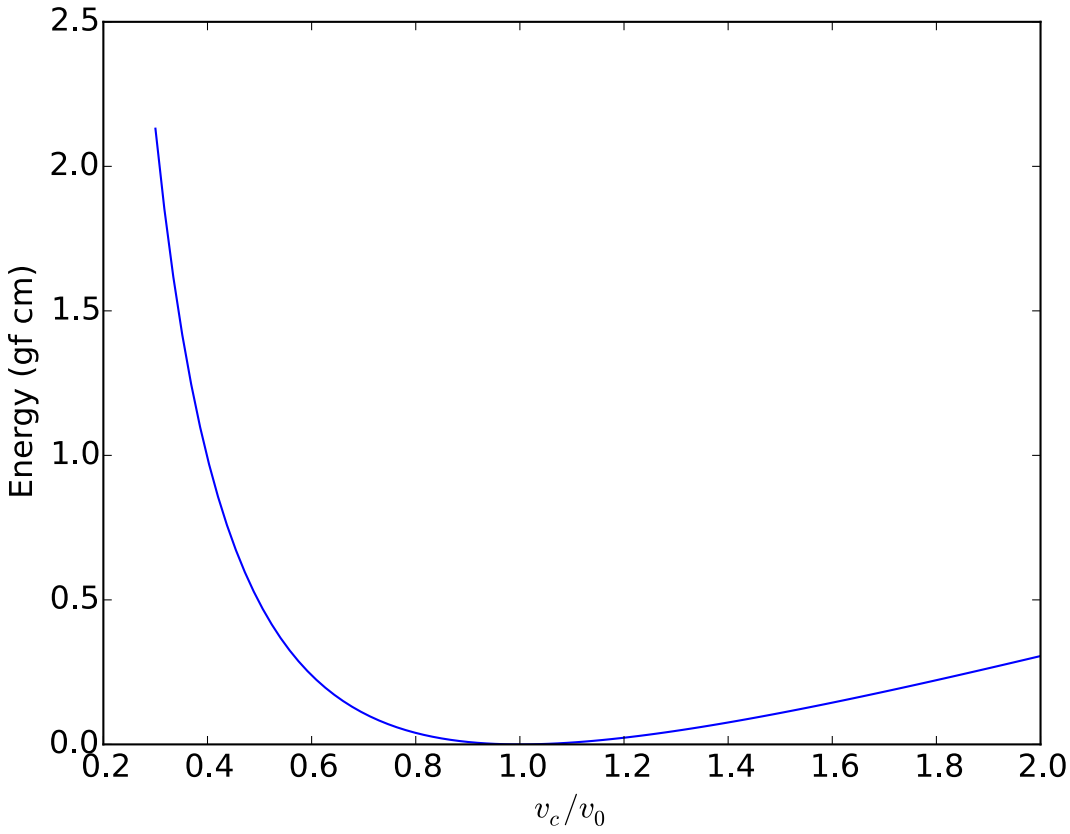


Figure 3.7: The evolution of strain energy within a van Wyk random fibre assembly.

Chapter 4

Knoppy Web Model

4.1 Introduction

In this chapter, the primary theoretical contribution of the thesis is presented: the first proposed mathematical model describing the compression mechanics of a knoppy web. It is the first step towards understanding the mechanics of fibrous assemblies containing spherical occlusions, and has been designed keeping in mind the hypotheses outlined in section 1.4.

At the outset, a disclaimer: the model is not intended to be a measuring stick for the performance of actual product samples. The assumptions that we make are necessary to develop this first simple model, but compromise its ability to provide quantitative results. Even if such results were reasonable, the inherent variability and manufacturing tolerances in the production process make any direct comparison precarious. The intent of this model is to provide an understanding of the roles that the various components play in the overall behaviour of knoppy web, and a qualitative indication of the effect that changes in the various parameters will have on this behaviour.

Section 4.2 presents the simple model of a spherical knop, which forms the core component of the overall model. Section 4.3 then incorporates this core into a model of a knoppy web "unit cell," combining it with the van Wyk [67] model of random fibre assemblies.

4.2 Simple sphere model

We have chosen to generalise the knop as a spherical membrane, developing the model from first principles using the energy method. In doing so, we use physical intuition, along with parallels with other fibrous systems, to make key assumptions that modify the behaviour of the spherical membrane to better approximate a knop.

We start by making several simplifying assumptions about the problem domain. Following on from the reasoning outlined in section 3.2.3, our first assumptions are as follows:

Assumption 4.2.1. *The sphere is a thin, hollow shell.*

Assumption 4.2.2. *The sphere is made of a continuous elastic membrane.*

Assumption 4.2.3. *The sphere is compressed axially between two parallel frictionless plates.*

Assumption 4.2.4. *The inherent properties of the knop, such as the thickness of the shell, do not change under compression.*

These assumptions differ little from a regular spherical shell model, and certainly over-simplify the mechanics of a knop—knops are neither continuous nor elastic, and generally tend to not be truly hollow. But subsequent assumptions will be made that will cause the model to behave less like a regular spherical shell and more like a knop. The first of these places a constraint on the geometry of a compessed knop:

Assumption 4.2.5. *During compression, the knop flattens against the plate, while maintaining a semi-circular edge profile.*

This assumption is made to prevent buckling, which as discussed in section 3.2.3 is present in regular spherical shells but not in knops.

Figure 4.1 shows the pre- and post-compression schematics for a sphere of initial radius r_0 and shell thickness $2h$, based on the above assumptions. We define c to be the distance that each plate has moved from its initial position, compressing the sphere by $2c$.

The compressed sphere can be divided into two regions: an inner cylindrical region of radius r' containing two flat circular membranes, and an outer edge region[1] with a radius of curvature of r''. By assumption 4.2.5, the semi-circular profile of the outer edge leads to the trivial relationship

$$r'' = r_0 - c$$

between r'' and c. Therefore, c and r' together uniquely define the shape of the compressed sphere, and form the primary parameters of the model.

By the energy method, the total energy developed within the sphere is equal to the sum of the independent energy terms. By assumption 4.2.2, we can calculate these terms using the definition of elastic strain energy per unit volume [76],

$$\mathrm{d}U_i = \frac{1}{2}\sum_j \sigma_{ij}\epsilon_{ij}\,\mathrm{d}V, \tag{4.1}$$

where σ_{ij} and ϵ_{ij} are the jth stress and strain components for the ith energy term, and $\mathrm{d}V$ is the infintesimal original volume to which the strain components are applied.

For the energy method to be applicable, it is a requirement that we select strain components that are in fact independent. To first order, this can be achieved by choosing strain components over orthogonal coordinates; the obvious choice is to consider in-plane membrane strain independent from bending strain. At large strains this delineation becomes less valid for true elastic membranes, as higher-order contributions (such as shear strain) lead to coupling between the strain components. However, because the wall of a knop is composed of discrete fibres rather than a continuous membrane, it is a reasonable to consider that the in-plane movement of the fibres would have a lower degree of coupling to their bending strain (even in the presence of the bonded PLA substructure). We therefore only consider first-order contributions to strain energy:

Assumption 4.2.6. *Higher-order strain terms are ignored.*

Since all strains are being applied to the uncompressed sphere, the best choice of coordinate system for the knop model is spherical polar coordinates. This provides the simplest means of expressing the relevant volumes. It also allows us to take advantage of the inherent symmetry about the axis of compression, to simplify later calculations. Thus, the equation for the ith energy term U_i will be of the form

$$U_i = \frac{1}{2}\int_\phi\int_\theta\int_r \sum_j \sigma_{ij}\epsilon_{ij}r^2\sin\theta\,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi. \tag{4.2}$$

The use of spherical coordinates does not inhibit our ability to use the model in later configurations, where the overall coordinate system is cartesian. By the energy method, we can use simple sums and geometric coupling to express the total energy of the configuration, regardless of what it might be composed of or how it is assembled.

We now move to assumptions about the individual regions. By assumption 4.2.6, the strain energy in each region can be split into membrane strain energy and bending strain energy.

As the sphere is compressed, the incremental changes in the inner region are flattening at the edges of the flat circular membranes, and in-plane membrane stretching/contraction.

In the outer region, strain contributions can be divided into two directions:

[1] The outer edge region is equivalent to the outer half of a torus.

- *About the axis of rotation:* The equatorial radii of points lie between r' and $r' + r''$. As the sphere is compressed, r'' decreases but r' increases[2]. Intuitively, the outer edge will go into tension around the axis, as it is forced outwards by the membrane strain within the inner region.
- *Along the median:* The geometry of the model prescribes that the radius of the median curve r'' will linearly decrease from r_0 towards zero[3] as the sphere is compressed.

Thus the incremental changes in the outer region under compression are an increase in tension about the axis (hoop strain), an increase in the equatorial radii of curvature, and bending of the meridian.

Bending strain energy is inversely proportional to the square of the radius of curvature. Hence, the strain energy developed in bending the meridian will be much larger than that developed in the small flattening at the edges of the inner region, or the small decrease in curvature around the axis. Therefore, we make the following first-order assumptions:

Assumption 4.2.7. *Bending energy can be ignored in the inner region.*

Assumption 4.2.8. *Bending energy can be ignored about the axis of revolution of the outer edge.*

Finally, we make the following first-order assumption to enable the mapping of points between the uncompressed and compressed spheres (presented in the next section):

Assumption 4.2.9. *Membrane energy can be ignored along the median of the outer edge.*

4.2.1 Mapping between the uncompressed and compressed spheres

We now turn our attention to the geometry of the sphere model. The assumptions from the previous section enable us to determine the behaviour of the model by calculating the various strain components within the sphere, and then calculating the elastic strain energy via eq. (4.2). Strain itself is a normalised measure of deformation, representing the relative displacements of the points in a medium from their initial positions. There is a one-to-one mapping between points on the uncompressed sphere and points on the sphere at any stage of compression; we can utilise this mapping to determine equations for the strain components within the sphere.

Figure 4.2 shows how the two regions on the compressed sphere map onto the uncompressed sphere. s'' is the meridian length along the midplane of the outer edge, ie.

$$\begin{aligned} s'' &= \frac{\pi}{2} r'' \\ &= \frac{\pi}{2}(r_0 - c). \end{aligned} \tag{4.3}$$

We define θ_c to be the polar angle on the uncompressed sphere that corresponds to the boundary between points that will lie in the inner region, and points that will lie in the outer edge.

The mapping between uncompressed and compressed states hinges on assumption 4.2.9. As the sphere is compressed, the movement of points within a particular region will depend on the strain components within that region.

- If the region contains a bending strain component, points on the midplane will not be altered by that strain component, but points inside of the midplane will be compressed and points outside the midplane will be stretched.
- If the region contains a membrane strain component, points in the same perpendicular plane to the strain direction will be compressed or stretched in the same manner.

[2] As noted earlier, r' is one of the primary parameters of the model, and is therefore afforded a range of values. However, as $r' = 0$ by definition on the uncompressed sphere, its value will always have increased.

[3] More precisely, assumption 4.2.4 mandates a geometric lower bound on r'' of h.

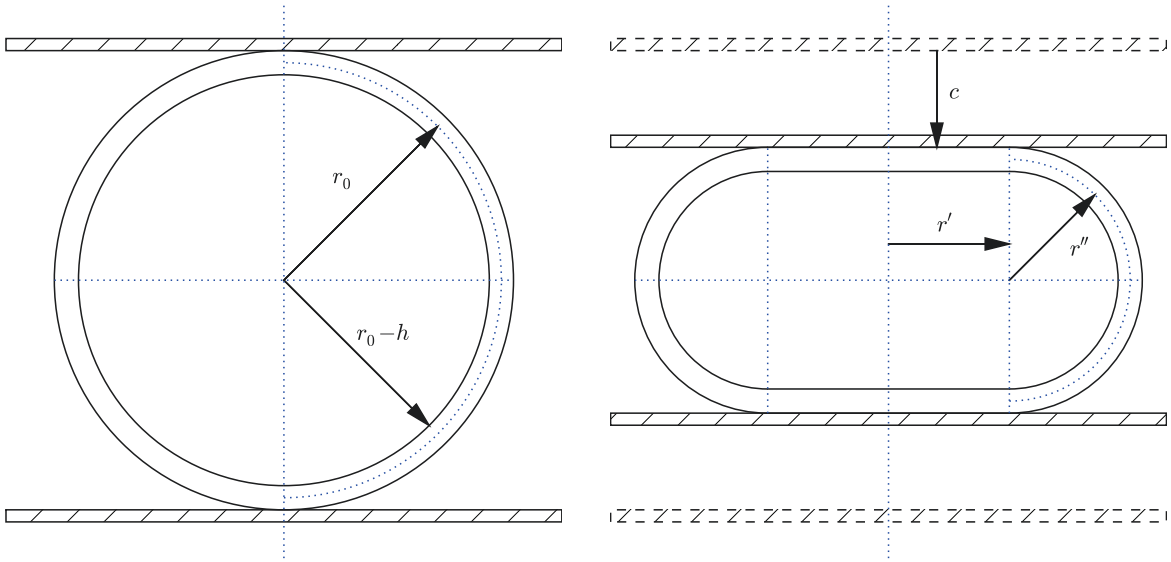


Figure 4.1: Schematics before and after compression for a knop of radius r_0 and shell thickness $2h$.

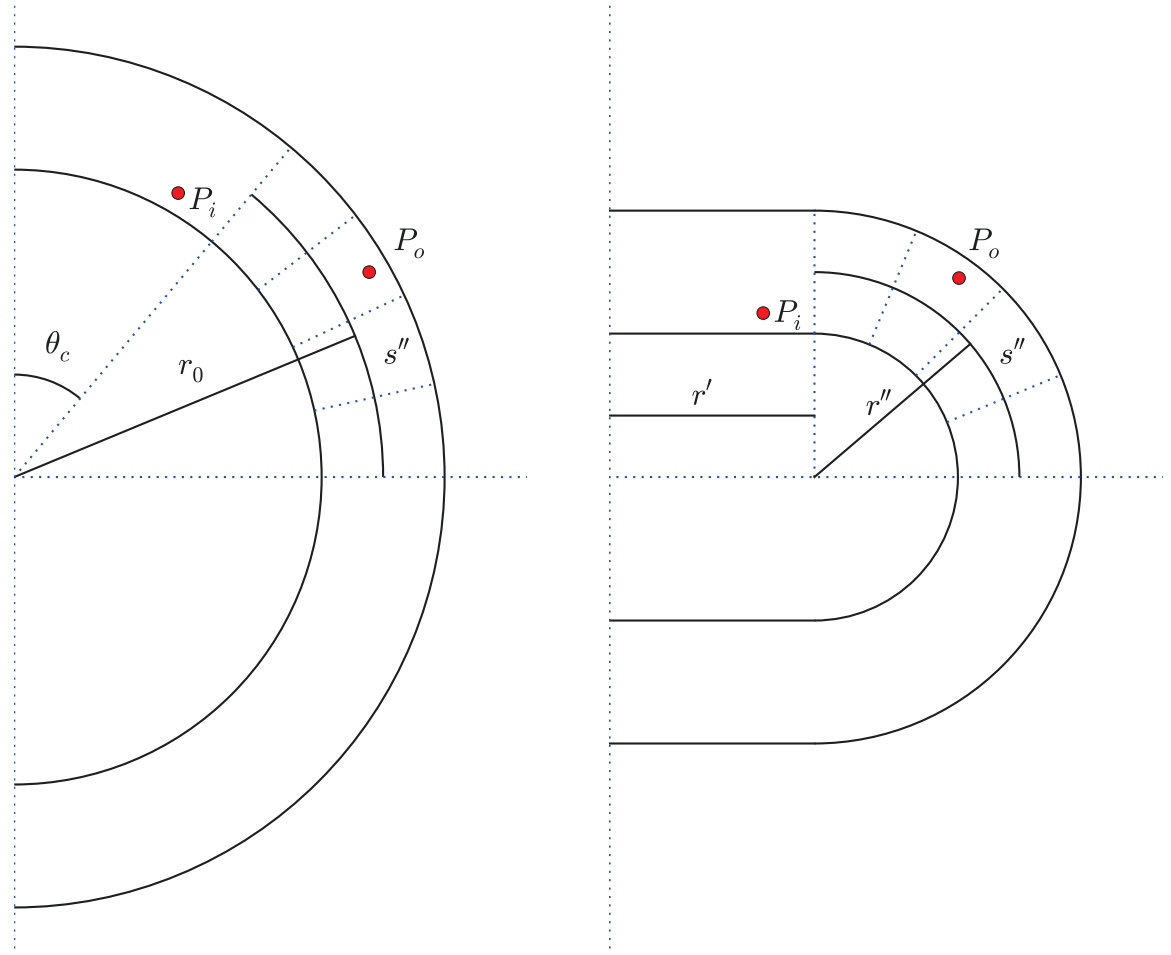


Figure 4.2: General mapping of points between the uncompressed sphere and the compressed sphere.

If membrane strain along the outer edge meridian is ignored (per assumption 4.2.9), then by definition the midplane length s'' must stay constant across the mapping. Therefore, the boundary on the uncompressed sphere defined by θ_c must be a distance s'' up from the equator along the midplane, as indicated in fig. 4.2. From this we see that

$$s'' = \left(\frac{\pi}{2} - \theta_c\right) r_0, \tag{4.4}$$

and therefore we can determine that

$$\begin{aligned} \theta_c &= \frac{\pi}{2} - \frac{s''}{r_0} \\ &= \frac{\pi}{2} - \frac{\pi}{2}\frac{r_0 - c}{r_0} \\ &= \frac{\pi}{2}\left(1 - \frac{r_0 - c}{r_0}\right) \\ &= \frac{\pi}{2}\frac{c}{r_0}. \end{aligned} \tag{4.5}$$

Thus θ_c increases linearly from 0 towards $\pi/2$ as c increases, and the volume of the uncompressed sphere that the outer edge region maps to decreases. In essence, material is transferred from the outer edge region to the inner region as the sphere is compressed, and takes on a new strain energy.

4.2.2 Inner region

The inner region of the compressed sphere contains two identical flat circular membranes. By assumption 4.2.7, we are ignoring bending strain and can consider just the in-plane strain within these membranes. We intuitively expect to have two strain components, along the meridian and about the axis. The energy density associated with these strains must be integrated over the original volume, which as mentioned earlier is most easily represented in spherical polar coordinates (see eq. (4.2)). But given that by assumption 4.2.5 we are ignoring buckling, it makes more sense to derive the strains themselves in cylindrical polar coordinates, because the strains will lie within a flat plane. This is in line with the reasoning behind assumption 4.2.7.

The equations of strain and shear strain in cylindrical polar coordinates are [77]

$$\epsilon_\rho = \frac{\partial u_\rho}{\partial \rho} \tag{4.6}$$

$$\epsilon_\phi = \frac{1}{\rho}\frac{\partial v_\phi}{\partial \phi} + \frac{u_\rho}{\rho} \tag{4.7}$$

$$\epsilon_z = \frac{\partial w}{\partial z} \tag{4.8}$$

$$\gamma_{\rho\phi} = \frac{1}{\rho}\frac{\partial u_\rho}{\partial \phi} + \frac{\partial v_\phi}{\partial \rho} - \frac{v_\phi}{\rho} \tag{4.9}$$

$$\gamma_{\rho z} = \frac{\partial w}{\partial \rho} + \frac{\partial u_\rho}{\partial z} \tag{4.10}$$

$$\gamma_{\phi z} = \frac{1}{\rho}\frac{\partial w}{\partial \phi} + \frac{\partial v_\phi}{\partial z}, \tag{4.11}$$

where u_ρ and v_ϕ are the components of displacement in the ρ and ϕ directions, and w is the component of displacement in the z direction (identical to in Cartesian coordinates). The various terms in the strain and shear strain equations are analogous to their Cartesian coordinate counterparts, aside from the additional term in each of ϵ_ϕ and $\gamma_{\rho\phi}$; those terms arise because a radial displacement causes a strain in the ϕ direction, and a displacement in ϕ causes an in-plane shear strain.

At this point we can make some simplifications. The inner region is symmetric about the axis of revolution in terms of geometry, material properties, loading and boundary conditions. This means that any displacements and stresses within the inner region must be independent of ϕ, and v_ϕ must be zero—there is no twisting as the sphere is compressed. This simplifies the above strains and shear strains

to

$$\epsilon_\rho = \frac{\partial u_\rho}{\partial \rho} \tag{4.12}$$

$$\epsilon_\phi = \frac{u_\rho}{\rho} \tag{4.13}$$

$$\epsilon_z = \frac{\partial w}{\partial z} \tag{4.14}$$

$$\gamma_{\rho\phi} = 0 \tag{4.15}$$

$$\gamma_{\rho z} = \frac{\partial w}{\partial \rho} + \frac{\partial u_\rho}{\partial z} \tag{4.16}$$

$$\gamma_{\phi z} = 0 \tag{4.17}$$

Additionally, by assumptions 4.2.1, 4.2.4 and 4.2.6 the plane stress simplifications for thin plates can be made—namely, that ϵ_z, $\gamma_{\rho z}$ and $\gamma_{\phi z}$ can be ignored because their contributions are miniscule. This leaves us with

$$\epsilon_\rho = \frac{\partial u_\rho}{\partial \rho} \tag{4.18}$$

$$\epsilon_\phi = \frac{u_\rho}{\rho}, \tag{4.19}$$

as expected.

Both of the above strains represent how a point has moved from its initial position. At this stage, we take another step to make the model more closely represent a knop than a spherical shell: we use different initial positions for determining ϵ_ρ and ϵ_ϕ.

Figure 4.3 shows how a point P at position (r, θ) in the inner region of the compressed sphere maps back to the uncompressed sphere, where

$$\rho_i = r\sin(\theta). \tag{4.20}$$

ρ_i and ρ_f are the initial and final radii of P in cylindrical polar coordinates, and s is the distance along the midplane of the uncompressed sphere to a point perpendicular to P.

For defining ϵ_ϕ, we will simply use ρ_i and ρ_f to represent the movement of P. This reflects the fact that fibres circling the axis will experience hoop strain as soon as they are deviated from their initial positions (see section 4.2.3.1 for further discussion on this point). However, fibres aligned along a meridian behave differently, and so ϵ_ρ must be handled accordingly.

Recall that by assumption 4.2.7, bending strain is ignored in the inner region. From a radial strain perspective, this effectively means that any stretching or compression occurs within a flat plane. Individual fibres won't experience in-plane strains simply from being unbent, and because there is much less coupling between layers of fibres within the wall of the knop than there is within a solid elastic membrane, the knop wall will experience negligible in-plane radial strain caused by bending. Therefore, when determining ϵ_ρ we use s as the initial radial position of P. If $u_s(P)$ is the in-plane displacement of P given the above considerations, then eq. (4.18) can be rewritten as

$$\epsilon_\rho = \frac{\partial u_s(P)}{\partial s}. \tag{4.21}$$

Recognising that on the flat circle the radius is equivalent to the surface distance, we can write down the final radius of the point after displacement as

$$\rho_f = s + u_s. \tag{4.22}$$

To progress further, we need to make an assumption about how the membrane behaves under compression, in order to relate points on the uncompressed sphere to points on the flattened inner region. We make the following assumption:

Assumption 4.2.10. *The radial strain within the inner region is constant throughout the membrane for any particular value of compression.*

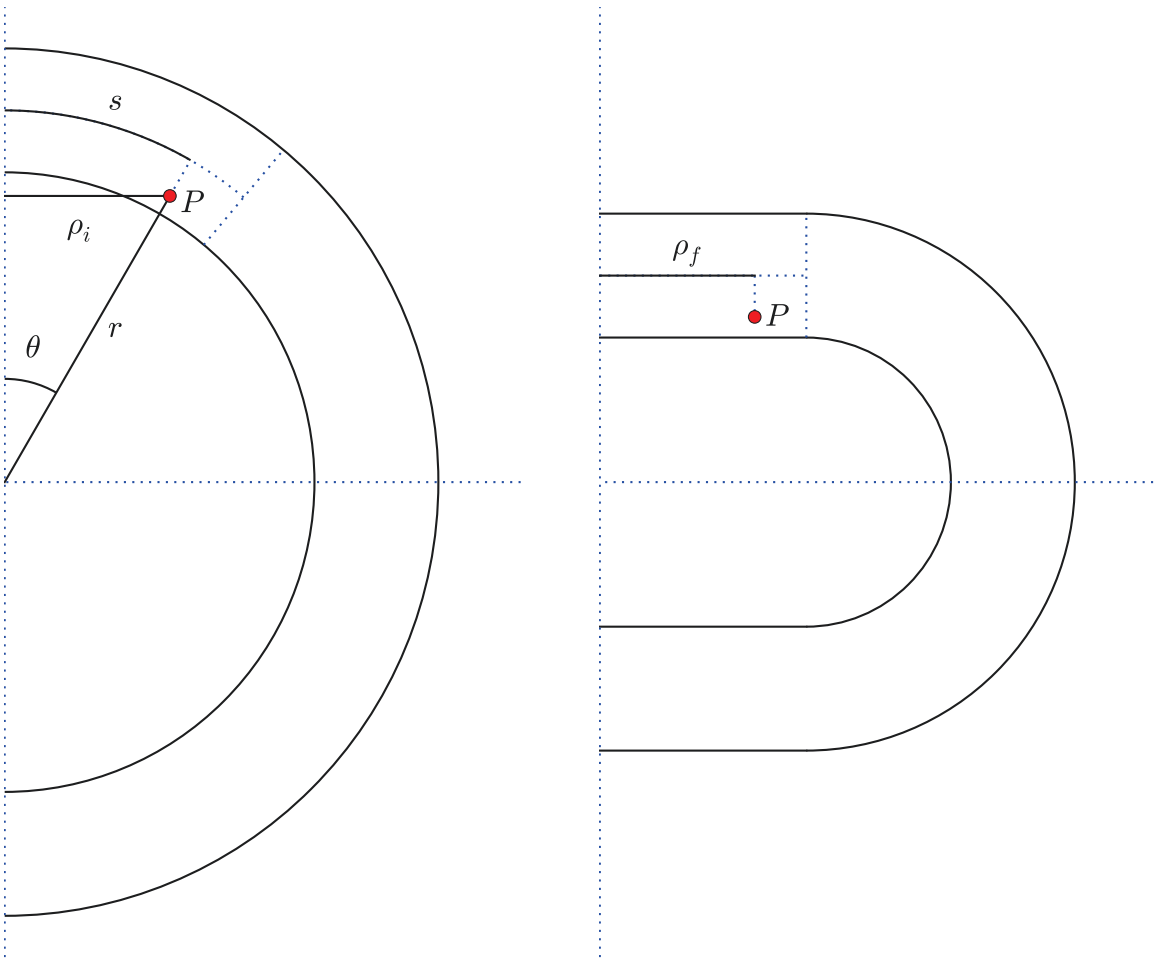


Figure 4.3: Mapping a point P between the uncompressed sphere and the inner region of the compressed sphere.

This assumption in combination with assumption 4.2.9 does create a strain discontinuity at the boundary between the inner and outer regions: the radial membrane strain drops from a constant value to zero instantly across the boundary. However, because we are using the energy method we can still calculate energy terms for the individual regions that make sense. The reasoning behind this assumption is that because the knop wall is more of a fibrous sheet than a continuous membrane, the radial strain within the inner region is more likely to accumulate within the plane of the flat circular membranes than to spread down over the curvature of the outer region. A more accurate representation might assume that the radial strain is zero at the equator and increases along the outer edge meridian to match up (somewhere) with the radial strain. This would also alter the way in which the outer edge minimised, lowering the hoop strain energy while creating a outer edge meridian membrane strain energy term. Alternatively, it could be assumed that there was a strain gradient across some small region in the vicinity of the boundary. However, either formulation would add considerable complexity to the way the membrane strain is specified, and increase the computing power required to run simulations. We have therefore chosen to ignore it for this simple model.

Using eq. (4.21),

$$\begin{aligned} u_s(P) &= \int_0^s \epsilon_\rho ds \\ &= s\epsilon_\rho, \end{aligned} \tag{4.23}$$

i.e. for a given compression, the displacement of a point with a midplane meridian distance s from the axis on the uncompressed sphere is equal to that distance multiplied by the radial strain. Let P' be a point on the region boundary, and let $s' = \theta_c r_0$ be the meridian length along the midplane on the uncompressed sphere to P'. Then $u_s(P') = r' - s'$, and substituting this into eq. (4.23), we find that

$$\begin{aligned} \epsilon_\rho &= \frac{u_s(P')}{s'} \\ &= \frac{r' - s'}{s'} \\ &= \frac{r'}{s'} - 1 \\ &= \frac{r'}{\theta_c r_0} - 1 \\ &= \frac{2}{\pi}\frac{r'}{c} - 1. \end{aligned} \tag{4.24}$$

To determine ϵ_ϕ, we substitute eq. (4.23) into eq. (4.22) to get

$$\begin{aligned} \rho_f &= s(1 + \epsilon_\rho) \\ &= r\theta(1 + \epsilon_\rho). \end{aligned} \tag{4.25}$$

Substituting $u_\rho = \rho_f - \rho_i$ into eq. (4.19),

$$\begin{aligned} \epsilon_\phi &= \frac{\rho_f - \rho_i}{\rho_i} \\ &= \frac{\rho_f}{\rho_i} - 1 \\ &= \frac{\theta}{\sin\theta}(1 + \epsilon_\rho) - 1. \end{aligned} \tag{4.26}$$

Note that for points close to the axis, $\sin\theta \approx \theta$ and $\epsilon_\phi \approx \epsilon_\rho$.

The constitutive law of a two-dimensional isotropic medium in polar coordinates is [78]

$$\sigma_\rho = \frac{E}{1-\mu^2}\left(\epsilon_\rho + \mu\epsilon_\phi\right) \tag{4.27}$$

$$\sigma_\phi = \frac{E}{1-\mu^2}\left(\epsilon_\phi + \mu\epsilon_\rho\right) \tag{4.28}$$

$$\tau_{\rho\phi} = \frac{E}{2(1+\mu)}\gamma_{\rho\phi}, \tag{4.29}$$

where E is the Young's modulus of the bar, and μ is Poisson's ratio. Therefore by eq. (4.1), the strain energy per unit volume is

$$\begin{aligned} dU &= \frac{1}{2}(\sigma_\rho \epsilon_\rho + \sigma_\phi \epsilon_\phi)\,\mathrm{d}V \\ &= \frac{E}{2(1-\mu^2)}\left[(\epsilon_\rho + \mu\epsilon_\phi)\epsilon_\rho + (\epsilon_\phi + \mu\epsilon_\rho)\epsilon_\phi\right]\,\mathrm{d}V \\ &= \frac{E}{2(1-\mu^2)}\left(\epsilon_\rho^2 + \epsilon_\phi^2 + 2\mu\epsilon_\rho\epsilon_\phi\right)\,\mathrm{d}V, \end{aligned} \tag{4.30}$$

and by eq. (4.2) the total strain energy is

$$\begin{aligned} U_{M_F} &= \frac{E}{2(1-\mu^2)}\int_\phi\int_\theta\int_r \left(\epsilon_\rho^2 + \epsilon_\phi^2 + 2\mu\epsilon_\rho\epsilon_\phi\right) r^2\sin\theta\,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi \\ &= \frac{E}{2(1-\mu^2)}\int_0^{2\pi}\int_0^{\theta_c}\int_{r_0-h}^{r_0+h} \left(\epsilon_\rho^2 + \epsilon_\phi^2 + 2\mu\epsilon_\rho\epsilon_\phi\right) r^2\sin\phi\,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi \\ &= \frac{\pi E}{1-\mu^2}\int_0^{\theta_c}\int_{r_0-h}^{r_0+h} \left(\epsilon_\rho^2 + \epsilon_\phi^2 + 2\mu\epsilon_\rho\epsilon_\phi\right) r^2\sin\phi\,\mathrm{d}r\,\mathrm{d}\theta. \end{aligned} \tag{4.31}$$

Substituting ϵ_ϕ into eq. (4.31),

$$\begin{aligned} U_{M_F} &= \frac{\pi E}{1-\mu^2}\int_0^{\theta_c}\int_{r_0-h}^{r_0+h}\left[\epsilon_\rho^2 + \left((1+\epsilon_\rho)\frac{\theta}{\sin\theta} - 1\right)^2 + 2\mu\epsilon_\rho\left((1+\epsilon_\rho)\frac{\theta}{\sin\theta} - 1\right)\right] r^2\sin\theta\,\mathrm{d}r\,\mathrm{d}\theta \\ &= \frac{\pi E}{1-\mu^2}\int_{r_0-h}^{r_0+h} r^2\,\mathrm{d}r\int_0^{\theta_c}\left[\epsilon_\rho^2 + (1+\epsilon_\rho)^2\frac{\theta^2}{\sin^2\theta} - 2(1+\epsilon_\rho)\frac{\theta}{\sin\theta} + 1 + 2\mu\epsilon_\rho\left((1+\epsilon_\rho)\frac{\theta}{\sin\theta} - 1\right)\right]\sin\theta\,\mathrm{d}\theta \\ &= \frac{\pi E}{1-\mu^2}\left[\frac{r^3}{3}\right]_{r_0-h}^{r_0+h}\int_0^{\theta_c}\left[\epsilon_\rho^2\sin\theta + (1+\epsilon_\rho)^2\frac{\theta^2}{\sin\theta} - 2(1+\epsilon_\rho)\theta + \sin\theta + 2\mu\epsilon_\rho(1+\epsilon_\rho)\theta - 2\mu\epsilon_\rho\sin\theta\right]\,\mathrm{d}\theta \\ &= \frac{2\pi E h(h^2+3r_0^2)}{3(1-\mu^2)}\int_0^{\theta_c}\left[(1+\epsilon_\rho)^2\frac{\theta^2}{\sin\theta} + (1+\epsilon_\rho^2 - 2\mu\epsilon_\rho)\sin\theta + 2(1+\epsilon_\rho)(\mu\epsilon_\rho - 1)\theta\right]\,\mathrm{d}\theta \\ &= \frac{2\pi E h(h^2+3r_0^2)}{3(1-\mu^2)}\left[(1+\epsilon_\rho)^2\int_0^{\theta_c}\frac{\theta^2}{\sin\theta}\,\mathrm{d}\theta + 2(1+\epsilon_\rho^2 - 2\mu\epsilon_\rho)\sin^2\left(\frac{\theta_c}{2}\right) + 2(1+\epsilon_\rho)(\mu\epsilon_\rho - 1)\frac{\theta_c^2}{2}\right]. \end{aligned} \tag{4.32}$$

4.2.3 Outer region

The outer region energy terms are conceptually much simpler than the inner region. As discussed at the beginning of section 4.2, there are only two strain components that need to be considered, and the geometry required to obtain them is more readily apparent.

Figure 4.4 shows how a point P on the outer edge of the compressed sphere maps back to the uncompressed sphere. r and θ are the radius and polar angle of P on the uncompressed sphere, as measured in spherical polar coordinates. $y = r - r_0$ is the distance from P to the midplane; by assumption 4.2.4, this stays constant as the sphere is compressed, and so the "spherical" radius of P on the compressed sphere is therefore $r - c$.

ρ_i and ρ_f are the uncompressed and compressed radii of P in cylindrical polar coordinates. We can derive equations for them, which will be beneficial to the hoop strain analysis in section 4.2.3.1. The initial horizontal radius of P is trivially

$$\rho_i = r\sin\theta. \tag{4.33}$$

To derive ρ_f, we first derive the angle θ_f from P to the equator on the compressed sphere. Following the same logic as we used to derive eq. (4.5), we define the length s along the midplane from the equator to (a point perpendicular to) P. Observing that by assumption 4.2.9 s is identical on both the uncompressed

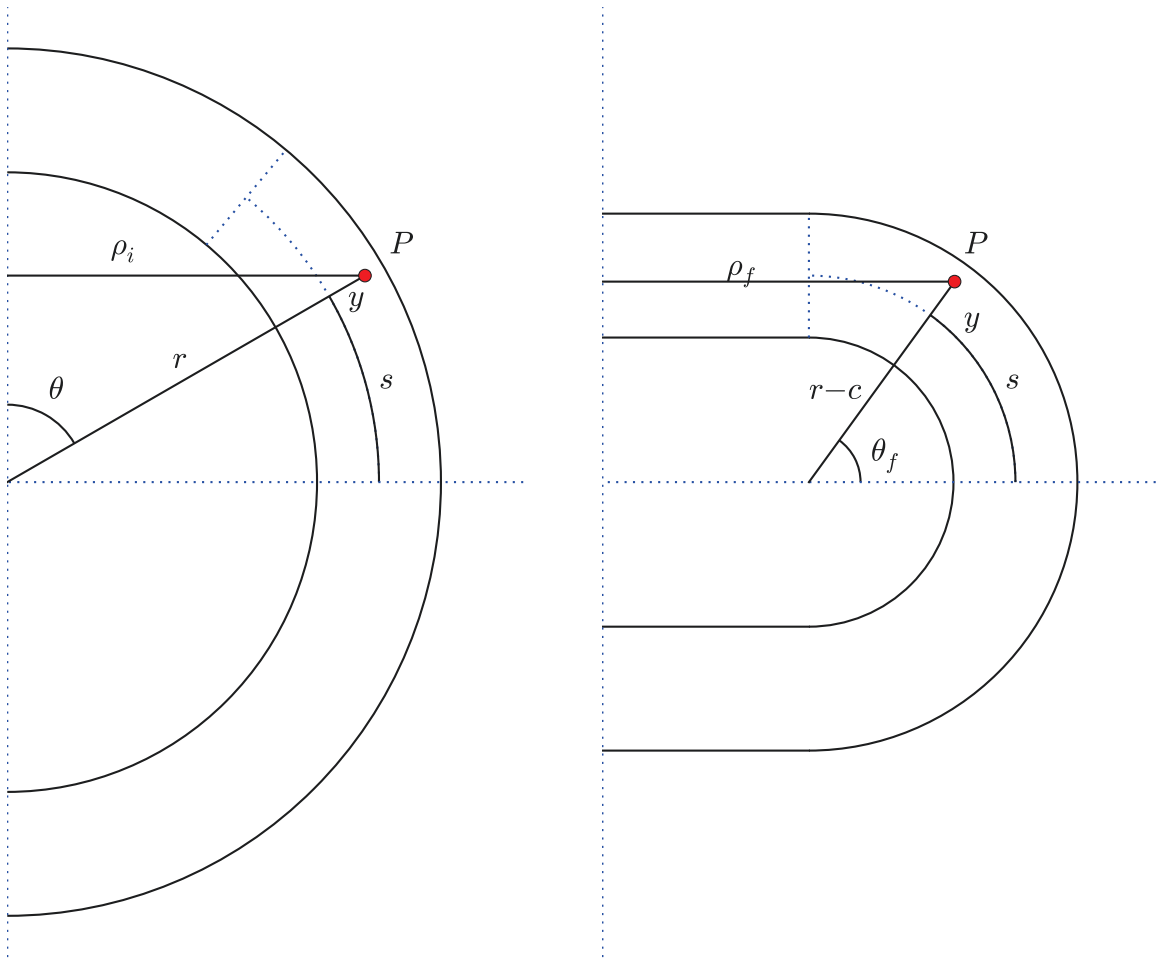


Figure 4.4: Mapping a point P between the uncompressed sphere and the outer edge region of the compressed sphere.

and compressed spheres, it follows that

$$\begin{aligned}\theta_f &= \frac{s}{r_0 - c} \\ &= \left(\frac{\pi}{2} - \theta\right) \frac{r_0}{r_0 - c}.\end{aligned} \tag{4.34}$$

Therefore, the horizontal radius of P on the compressed sphere is

$$\begin{aligned}\rho_f &= r' + (r - c)\cos(\theta_f) \\ &= r' + (r - c)\cos\left[\left(\frac{\pi}{2} - \theta\right) \frac{r_0}{r_0 - c}\right].\end{aligned} \tag{4.35}$$

We are ignoring bending energy around the axis by assumption 4.2.8, and membrane energy along the meridian by assumption 4.2.9. Thus we only need to consider two sources for strain energy developed within the outer ring: hoop strain around the axis, and bending strain along the meridian.

4.2.3.1 Membrane energy from hoop strain

An average standard knop may have a radius of anywhere from 0.25 cm to 0.5 cm, corresponding to a circumference of between 1.57 cm and 3.14 cm. The cutting length generally used in the production process is 55 mm (table 2.1), which is significantly longer. This leads to the conclusion that most fibres within a knop will curl around on themselves at least once, often more. The presence of the bonded PLA skeleton interwoven within the knop will in general inhibit any potential unravelling of the wool fibres. Thus, as far as membrane strain is concerned, when the knop is compressed the fibres going into tension can be likened to continuous hoops—at least up until the point that the tension force within the hoop exceeds the single fibre withdrawal force for a knop. Below that, the tension in the hoops can be considered an aggregate representation of the extension and crimp reduction of axial fibres, and of the deflection of neighbouring fibre contacts.

Per assumption 4.2.7, the membrane strain energy in the outer ring is purely axial. Intuitively, it is the fibres aligned around the axis that will contribute most to the membrane strain[4]. Therefore, we make the following assumption:

Assumption 4.2.11. *The equatorial strain energy density within the outer edge membrane is equivalent to that within a thin hoop under axial tension.*

This continues our generalisation of the knop as a spherical membrane.

A thin hoop under axial tension is mechanically equivalent to a thin bar being stretched. The strain in the hoop is simply

$$\epsilon_\phi = \frac{l_f}{l_i} - 1, \tag{4.36}$$

where l_i is the original hoop length (circumference) from the uncompressed sphere, and l_f is the current hoop length. To obtain ϵ_ϕ in terms of model parameters, we first recognise that

$$\begin{aligned}\epsilon_\phi &= \frac{l_f}{l_i} - 1 \\ &= \frac{2\pi\rho_f}{2\pi\rho_i} - 1 \\ &= \frac{\rho_f}{\rho_i} - 1.\end{aligned} \tag{4.37}$$

Substituting eq. (4.33) and eq. (4.35) into eq. (4.37),

$$\epsilon_\phi = \frac{r' + (r - c)\cos\left[\left(\frac{\pi}{2} - \theta\right)\frac{r_0}{r_0 - c}\right]}{r\sin\theta} - 1. \tag{4.38}$$

[4] For fibres that are at an angle to the axis, membrane strain will manifest as a shear strain within the fibres, which is ignored by assumption 4.2.6.

For an elastic thin bar, the stress is

$$\sigma_\phi = E\epsilon_\phi, \tag{4.39}$$

where E is the Young's modulus of the bar. Therefore by eq. (4.1), the strain energy per unit volume is

$$\begin{aligned} dU &= \frac{1}{2}\sigma_\phi\epsilon_\phi \,\mathrm{d}V \\ &= \frac{1}{2}E\epsilon_\phi^2 dV, \end{aligned} \tag{4.40}$$

and by eq. (4.2) the total strain energy is

$$\begin{aligned} U_{M_S} &= \frac{1}{2}E\int_\phi\int_\theta\int_r \epsilon_\phi^2 r^2\sin\theta \,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi \\ &= E\int_0^{2\pi}\int_{\theta_c}^{\frac{\pi}{2}}\int_{r_0-h}^{r_0+h} \epsilon_\phi^2 r^2\sin\theta \,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi \\ &= 2\pi E\int_{\theta_c}^{\frac{\pi}{2}}\int_{r_0-h}^{r_0+h} \epsilon_\phi^2 r^2\sin\theta \,\mathrm{d}r\,\mathrm{d}\theta. \end{aligned} \tag{4.41}$$

4.2.3.2 Bending energy along the meridian

The bending process in the outer region of the compressed spherical knop is similar in nature to bending a two-dimensional random fibre assembly, in that the fibres which contribute the most to the bending energy will be those aligned along the direction of bending[5]. The contributions will also change across the thickness of the sphere wall; fibres closer to the inside will have smaller initial and final curvatures.

Assumption 4.2.12. *The bending strain energy density within the outer edge membrane is equivalent to that within a thin curved bar being acted upon by a pure moment.*

For a thin curved bar being acted upon by a pure moment, the bending strain at a point within the bar is

$$\epsilon = -(\kappa_2 - \kappa_1)y, \tag{4.42}$$

where κ_1 is the initial curvature of the bar, κ_2 is the final curvature, and y is the perpendicular distance of the point from the neutral plane along the centre of the bar. By geometric analogy with fig. 4.4, we observe that $\kappa_1 = 1/r_0$, $\kappa_2 = 1/(r_0 - c)$ and $y = r - r_0$.

By the common elements between assumption 4.2.11 and assumption 4.2.12, eq. (4.39) also applies here. Therefore by eq. (4.1), the bending strain energy per unit volume is

$$\begin{aligned} dU_B &= \frac{1}{2}\sigma\epsilon \,\mathrm{d}V \\ &= \frac{1}{2}E\epsilon^2 \,\mathrm{d}V \\ &= \frac{1}{2}E(\kappa_1 - \kappa_2)^2 y^2 \,\mathrm{d}V \\ &= \frac{1}{2}E(\kappa_1 - \kappa_2)^2 (r - r_0)^2 \,\mathrm{d}V, \end{aligned} \tag{4.43}$$

[5]This can be reasoned by simple geometry: bending a fibre involves bringing the ends of the fibre closer together. If the fibre is lying at an angle to the direction of bending, then only a component of the motion will actually bring the fibre ends closer together. The rest of the bending motion will go into twisting the fibre, which is ignored by assumption 4.2.6.

and by eq. (4.2) the total bending strain energy is

$$\begin{aligned} U_B &= \frac{1}{2}E(\kappa_1-\kappa_2)^2\int_\theta\int_\phi\int_r (r-r_0)^2 r^2\sin\theta\,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi \\ &= E(\kappa_1-\kappa_2)^2\int_0^{2\pi}\int_{\theta_c}^{\frac{\pi}{2}}\int_{r_0-h}^{r_0+h}(r-r_0)^2r^2\sin\theta\,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi \\ &= 2\pi E\left(\frac{1}{r_0}-\frac{1}{r_0-c}\right)^2\int_{\theta_c}^{\frac{\pi}{2}}\sin\theta\,\mathrm{d}\theta\int_{r_0-h}^{r_0+h}(r-r_0)^2r^2\,\mathrm{d}r \\ &= 2\pi E\left(\frac{1}{r_0}-\frac{1}{r_0-c}\right)^2\cos(\theta_c)\int_{r_0-h}^{r_0+h}(r-r_0)^2r^2\,\mathrm{d}r \\ &= 4\pi E\left[\frac{h^3}{3}r_0^2+\frac{h^5}{5}\right]\left(\frac{1}{r_0}-\frac{1}{r_0-c}\right)^2\cos(\theta_c)\,. \end{aligned} \tag{4.44}$$

4.2.4 Total energy

By the energy method, the total energy of the system is the sum of the various energy contributions, ie.

$$U_T = 2U_{M_F} + U_{M_S} + U_B, \tag{4.45}$$

where U_{M_F}, U_{M_S} and U_B are the energy terms from the preceding sections. U_{M_F} must be counted twice to account for the presence of both the top and circular membranes within the inner region. The equations for U_{M_S} and U_B already include the necessary factor of 2 to cover their full regions.

The overall knop model has four geometric parameters:

- c, the amount by which the knop has been compressed.
- r', the radius of the inner region in cylindrical polar coordinates.
- r_0, the radius of the uncompressed knop in spherical polar coordinates.
- $2h$, the thickness of the knop wall.

Additionally, there are two material parameters:

- E_k, the Young's modulus of the knop membrane.
- μ_k, the Poisson's ratio of the knop membrane ($\equiv \mu$ in eq. (4.32)).

r_0, h, E_k and μ_k are properties of the knop, and, by assumption 4.2.4, are considered to be constants throughout the simulation. c and r' therefore become the simulation parameters, over which the energy is calculated. Of these two, c is the independent variable, that we "control" by squashing the knop. To specify r', we use the minimum energy principle: the value of r' for any given c will be that which results in the lowest U_T.

Each of the three energy components specified a Young's modulus E. In defining E_k we are making the following assumption:

Assumption 4.2.13. *The Young's moduli within each of the three energy terms are equal.*

Given that the orientation of knops within a knoppy web is mostly random, this is a fair simplifying assumption. More importantly, we are defining E_k as an aggregate property of the knop that defines its "stiffness." E_k therefore acts as a uniform scaling factor on the energy of the knop. Several processing parameters exist for altering the properties of a knop (as alluded to in section 2.6); we are using E_k to represent the sum total of these effects.

Table 4.1: Sphere model parameters

Parameter	Units	Number of steps	Value
c	cm	360	$[0.001, r_0 - h]$
r'	cm	360	$[0.001, 2r_0]$
r_0	cm		0.25
h	µm		50
E_k	gf/cm^2		5.1 x 10^5
μ_k			0.5

4.2.5 Analysis

Table 4.1 shows the units of the various parameters in the model, and the values used for analysis. The units chosen are intentionally not SI units (which would be Pascals and metres); instead, the units are chosen to be consistent with common units used in textile mechanics papers (e.g. [70, 72, 79]). In particular, energies will be given in gf cm to be consistent with other energy method papers for fibrous assemblies (e.g. [72][6]).

The lower limit for both c and r' was set just above zero to avoid divide-by-zero errors. The upper limit for r', meanwhile, was made sufficiently high to ensure that subsequent minimisation would give the correct minima, while not unduly compromising the run time of the model.

The upper limit for c was set to the absolute geometric limit for the purposes of exploring the full parameter space; however, behaviour at very high c should be treated with caution, due to the limits of the classical bar bending theory utilised in section 4.2.3.2. A soft upper limit of $c = (3/4)(r_0 - h)$ restricts the curvature increase of the outer edge to a maximum of $\kappa_2/\kappa_1 = 4$.

h was chosen to be on the order of a few fibre thicknesses. E_k and μ_k were chosen somewhat arbitrarily for this behavioural analysis[7]; given the unusual nature of this model, there is no good reference point for their values. They will be the fitting parameters when the model is fitted to experimental data in section 5.5.

The simple sphere model was implemented in Python 2.7 [82], using the IPython interactive computing environment [83]. The symbolic mathematics library SymPy [84] was used to construct the equations, and numerical integration was performed using the NumPy [85] and SciPy [86] libraries. IPython's parallel computing toolkit enabled the use of multiple processing cores to calculate U_T across thousands of combinations of c and r', and its interactive tools were used to tangibly visualise the compression behaviour of the model. A full code listing is provided in appendices B.1 and B.2.

4.2.5.1 Total energy

Figure 4.5 shows the behaviour of U_T with compression, for varying values of horizontal spread. There is a clear and stable minimum that lies between the two plotted boundary lines, which matches our intuition:

- If the sphere were constrained such that it could not expand (and there was zero hoop strain at the outermost edge), then by section 4.2 it follows that $r' = c$. The minimum lies above this, showing that the favourable energy configuration involves some spread, and that the hoop strain in the outer edge is positive (in tension).
- If the inner region were radially uncompressed, then $\epsilon_\rho = 0$ and by eq. (4.24) it follows that $r' = \frac{\pi}{2}c$. The minimum lies below this, indicating that the inner region is radially compressed; thus the hoop strain terms in both the inner and outer regions (eqs. (4.26) and (4.38)) are inhibiting spread.

[6] Note that in [72] energies are presented in mN cm, but their computational code uses gf cm internally along with the approximation 1 gf $\approx$ 1 cN.

[7] Specifically, we initially chose $E_k = 0.05$ GPa and $\mu_k = 0.5$ as being close to the typical values for rubber [80, 81].

By minimising along the r' axis, we can obtain a more familiar representation of compression behaviour. Figure 4.6 shows the minimum total energy of the sphere as it is compressed. This curve shows far more clearly that the model is behaving as we would intuitively expect: the energy within the sphere increases as it is compressed. Interestingly, the strain energy does not increase uniformly, but in fact tapers off after about 60% compression, before spiking at extreme compression.

4.2.5.2 Energy components

Now we delve into the behaviour of the individual energy components. We are particularly interested in observing which energy terms dominate at which stages of compression, but it is also important to confirm that each energy term is behaving as we would expect based on physical intuition.

Figure 4.7 shows the behaviour of U_{M_F} as the sphere is compressed. There is a minimum that approximately follows the line $r' = \frac{\pi}{2}c$, lagging underneath it at large c. This fits with the expected behaviour of the inner region in isolation. The line corresponds to setting $\epsilon_\rho = 0$ in eq. (4.24), and so the preferred configuration for the inner region is to have no radial strain. However, from section 4.2.2 the radial strain does not depend on the "flattening" of the inner region, whereas the hoop strain does; thus at large c, increasing r' to account for the flattening will increase ϵ_ϕ, creating an opposing force inwards. This leads to the observed small negative ϵ_ρ.

Figure 4.8 shows the behaviour of U_{M_S} as the sphere is compressed. Hoop strain depends solely on how far the outer edge is deviated from its uncompressed position. The two boundary lines correspond to the two possible extremes:

- The radial position of the outer edge of the outer region remaining unchanged ($r' = c$).
- The radial position of the boundary between the inner and outer regions remaining unchanged ($r' = r_0 \sin(\theta_c)$).

The U_{M_S} minimum lies between these boundaries, as expected.

Unlike U_{M_F}, U_{M_S} has a convex shape along its minimum, decreasing in energy at high c. This is due to the shifting of the θ_c boundary between the inner and outer regions. As the outer edge bends, points on the outside of the sphere will move inwards relative to the midplane, and points on the inside of the sphere will move outward. Thus once compression begins, there will always be some position where the outer edge is under some hoop strain. However, as $c \to r_0$ the volume of the outer region decreases towards zero, and it becomes progressively easier to obtain a value of r' that places all points within the outer region near their uncompressed positions.

Figure 4.9 shows the behaviour of U_B as the sphere is compressed. The energy increases approximately exponentially as c increases, which fits with our physical intuition about the model: we expect the increasing curvature difference to dominate over the decrease in volume of fibre contributing to U_B.

At each value of c, the minimum U_T plotted in fig. 4.6 was found by minimizing along the r' axis. By recording the value of r' corresponding to the minimum U_T, we can read off from our model the values of the various energy components at that (c, r') location. Figure 4.10 shows the various components of the minimum strain energy from fig. 4.6. It is clear that the U_{M_S} term dominates the behaviour at early compression, and also gives rise to the observed tapering off at about 60% compression in fig. 4.6. At high compression, U_{M_S} drops out and the U_{M_F} term dominates.

Also obvious is that the U_B term contributes very little to the overall strain energy for the majority of compression. This does not mean that it should have been ignored outright—after all, it is the component that is increasing most rapidly for much of the compression range. It becomes particularly massive above 90% compression (visible in section 4.2.5.2), in keeping with the general behaviour in fig. 4.9. But at the strain region where it begins to become significant in magnitude, the ratio of curvatures is well above the previously-indicated limits for this term to remain valid. The bending strain energy built up in the

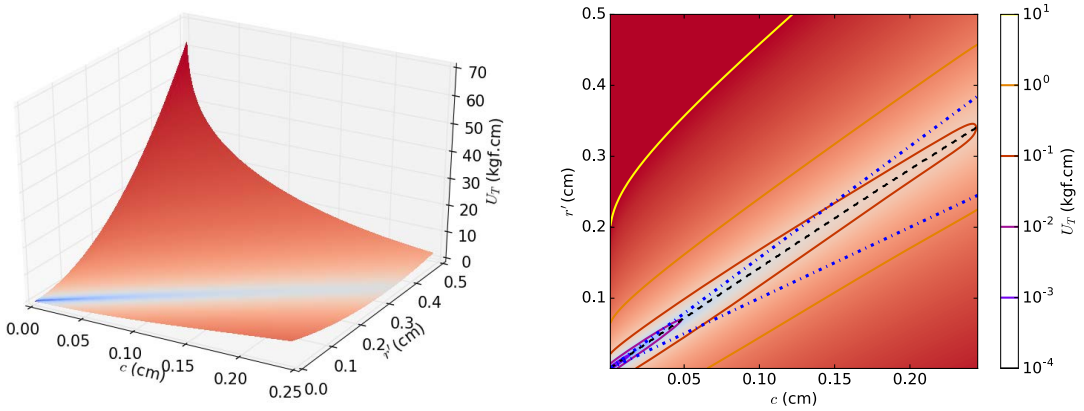


Figure 4.5: Surface and contour plots of U_T. The lower blue line on the contour plot is $r' = c$, and the upper blue line is $r' = \frac{\pi}{2}c$.

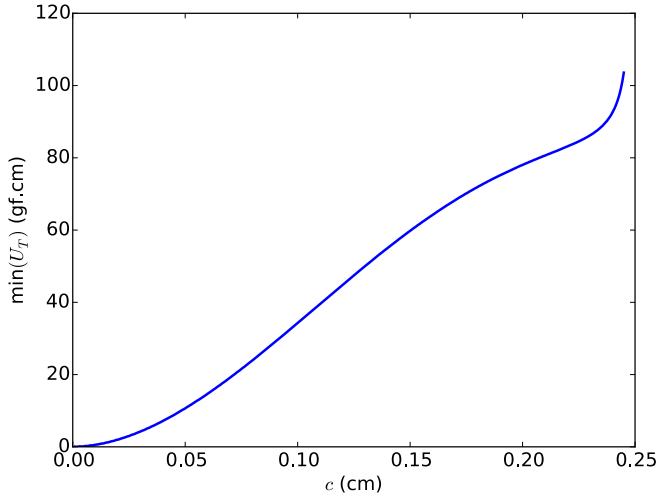


Figure 4.6: Strain energy of a sphere under compression.

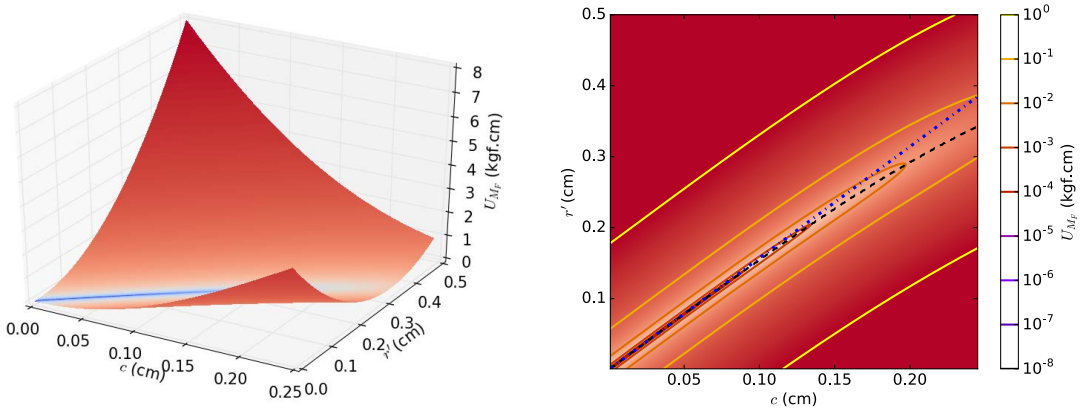


Figure 4.7: Surface and contour plots of U_{M_F}. The blue line is $r' = \frac{\pi}{2}c$.

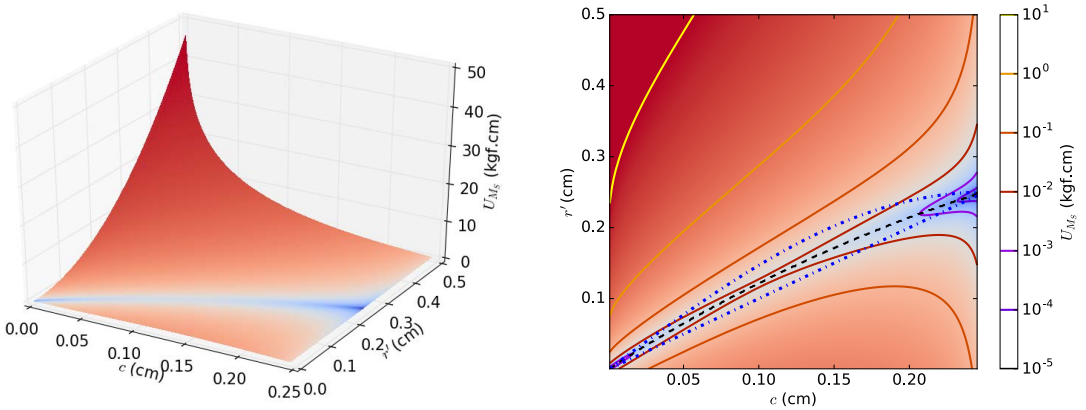


Figure 4.8: Surface and contour plots of U_{M_S}. The lower blue line on the contour plot is $r' = c$, and the upper blue line is $r' = r_0 \sin(\theta_c)$.

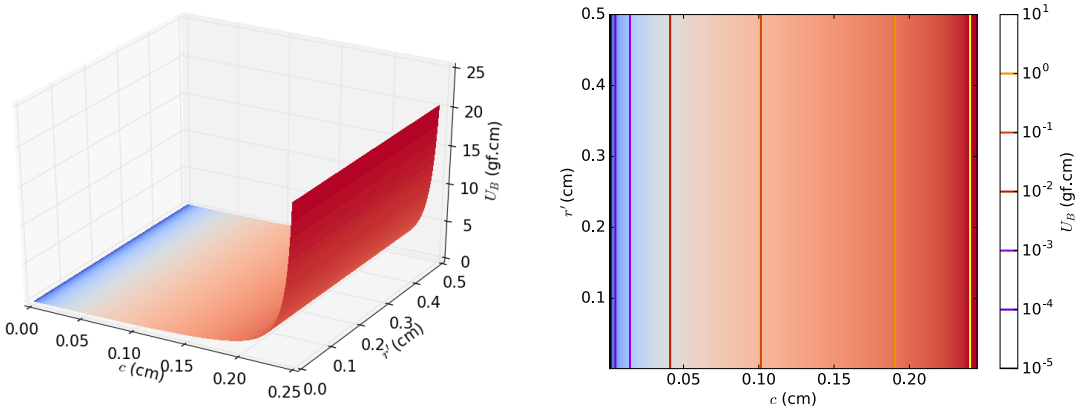
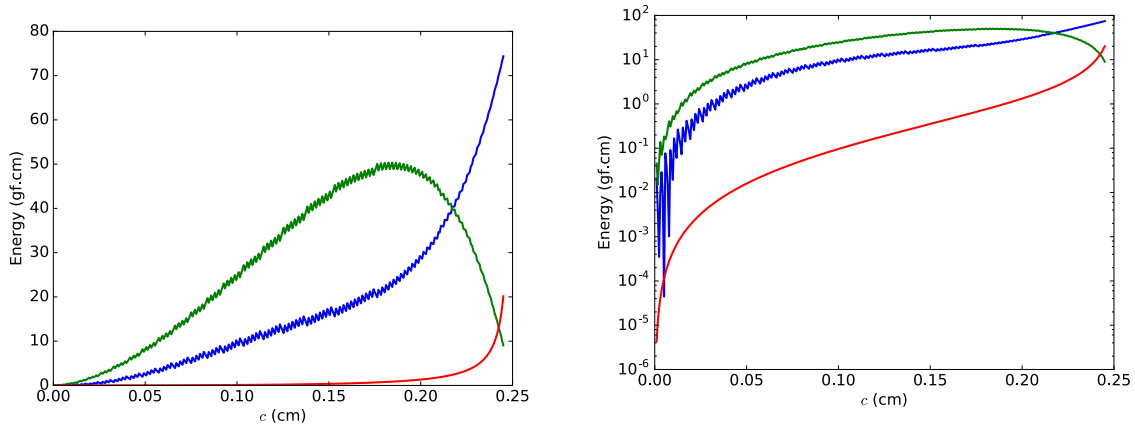


Figure 4.9: Surface and contour plots of U_B.



(a) (b)

Figure 4.10: Energy components of a sphere under compression, plotted on (a) normal and (b) log-linear scales. Blue is $2U_{M_F}$, green is U_{M_S} and red is U_B.

outer edge is simply not comparable in magnitude to the other components. This is counter-intuitive, as we would naturally assume that knops get most of their strength from the bending strain around their equator. Mathematically, it is essentially a consequence of assumptions 4.2.2 and 4.2.11, which enable most of the strain to be contained in the model within the membrane strains. Physically, one must remember that in our fibrous knop spheres, "membrane strain" still involves the bending of fibre segments, and there are many more fibre segments within the plane of the inner region than there are lying perpendicular across the equator.

It is mildly interesting to note that there is underlying instability in the U_{M_F} and U_{M_S} terms. This is simply a numerical artifact of the integration and minimisation process. U_{M_F} and U_{M_S} both have a dependence on both c and r', and the size of the instabilities is directly linked to the number of steps taken. Their instabilities exactly cancel each other out, leaving U_T stable. The U_B energy term has no relationship to r', because the curvature of the outer edge is geometrically linked to c alone.

4.2.5.3 Effect of parameters

There are four parameters in this model: the knop radius r_0, the knop wall thickness h, the Young's modulus E_k of the knop wall, and the Poisson's ratio μ_k of the knop wall (section 4.2.4).

Mathematically, E_k is simply a global scaling factor; it will trivially increase the strain energy of all components, and we therefore do not show its behaviour here. Physically, E_k is a measure of the stiffness of a sheet of fibres. In theory it would be possible to predict its value based on the micromechanics of the fibre sheet, similar to what has been done for regular fibrous assemblies [70]. However, the main issue that one runs into is the fact that a knop wall is highly anisotropic, and the knop itself highly discretised; thus many of the assumptions and techniques used in the existing micromechanical derivations would not be applicable. A derivation of this kind would go hand-in-hand with an in-depth study of the knop formation process. We leave this to future workers.

Figure 4.11 shows the effect of varying r_0 on the minimised strain energy. Increasing the size of the sphere scales up all energy terms, but doesn't change their shape significantly.

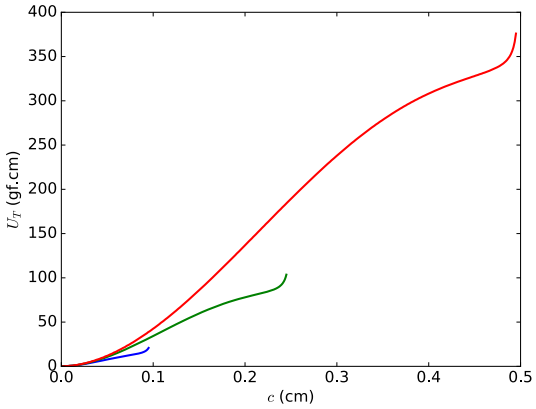
Figure 4.12 shows the effect of varying h on the minimised strain energy. Unlike E_k, the energy terms are not linearly proportional to h; yet still, the strain energy increase is only slightly higher than linear for the most part. At extreme strains there is a significant increase in U_B, which is expected given the h^5 and h^3 terms in eq. (4.44).

Figure 4.13 shows the effect of varying μ_k on the minimised strain energy. U_B is unaffected (fig. 4.13d) because μ_k is only present in eq. (4.32), and U_B has no dependence on r'. Thus the effect of increasing μ_k is to suppress U_{M_F} and increase U_{M_S} (figs. 4.13b and 4.13c). This enables tuning of the magnitude of the higher-strain convex behaviour.

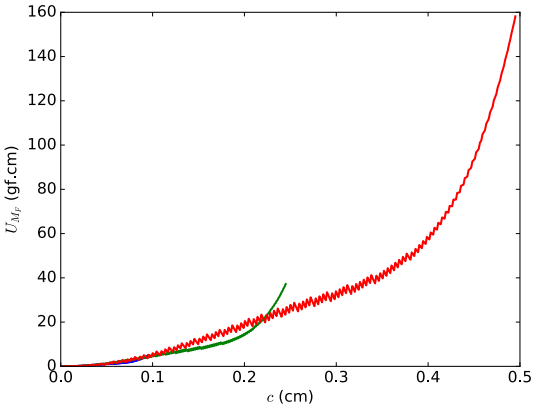
4.3 Knoppy web unit cell

Armed with a model describing the behaviour of a knop under compression, the next step is to combine it with an existing model of random fibre assemblies to create a "unit cell" of knoppy web. Defining a unit cell requires assumptions about the underlying structure. If the knoppy web blend is sufficiently blended, the knops will be positioned relatively randomly across the knoppy web. However, there is generally a gradient in density of knops across the thickness of the knoppy web (as discussed in section 2.7), and the knops are generally of a comparable size to the thickness of the knoppy web. Thus it is not reasonable to rely on isotropic assumptions about the positioning of knops. At the same time, we do need to make some simplifying assumption in order to define a repeatable unit cell. For the purpose of this simple model, we therefore make the following assumption:

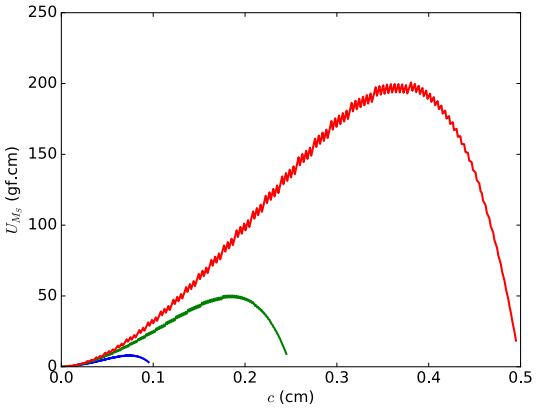
Assumption 4.3.1. *The knops are arranged within the knoppy web in a regular three-dimensional matrix.*



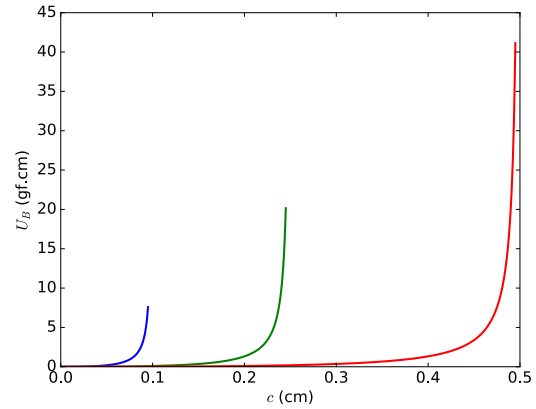
(a) From bottom, $r_0 = [0.1, 0.25, 0.5]$cm



(b) Corresponding values of U_{M_F}

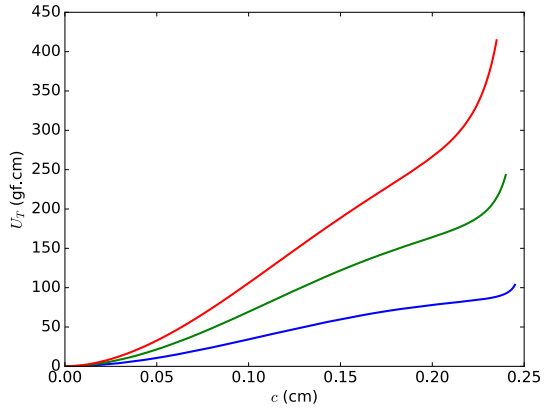


(c) Corresponding values of U_{M_S}

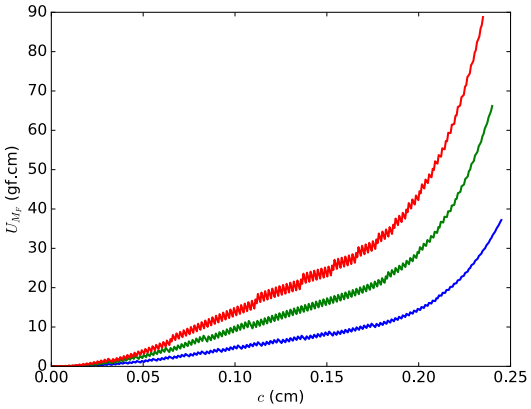


(d) Corresponding values of U_B

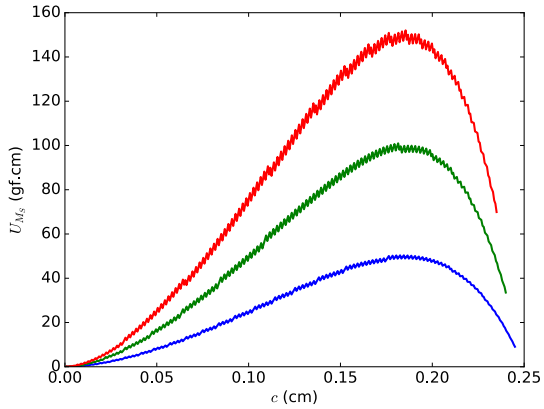
Figure 4.11: Minimised sphere energy for a range of r_0.



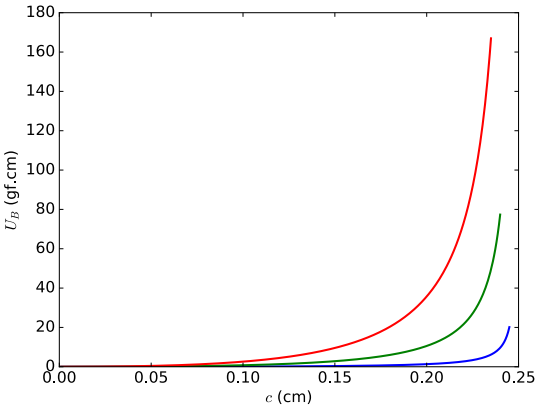
(a) From bottom, $h = [50, 100, 150]$µm



(b) Corresponding values of U_{M_F}

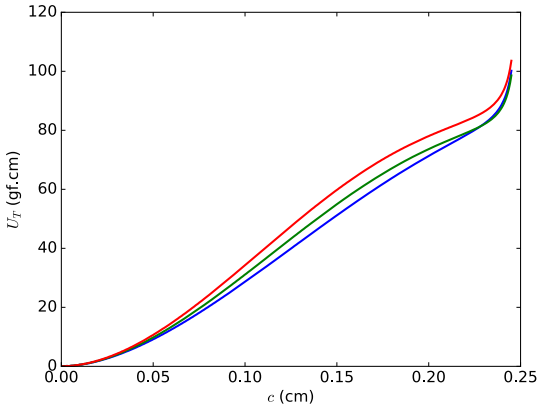


(c) Corresponding values of U_{M_S}

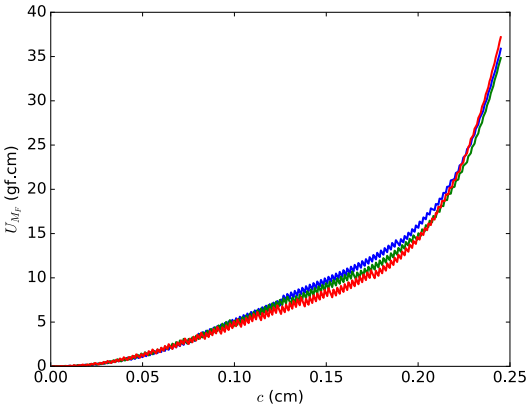


(d) Corresponding values of U_B

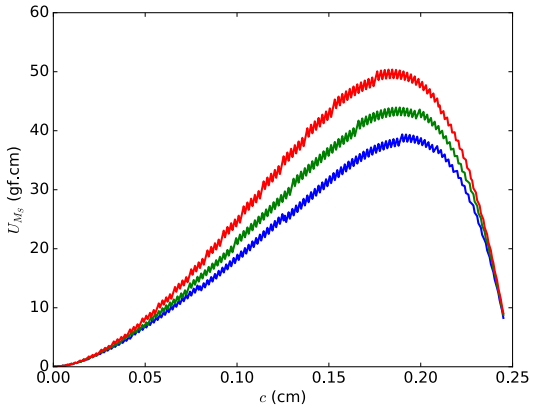
Figure 4.12: Minimised sphere energy for a range of h.



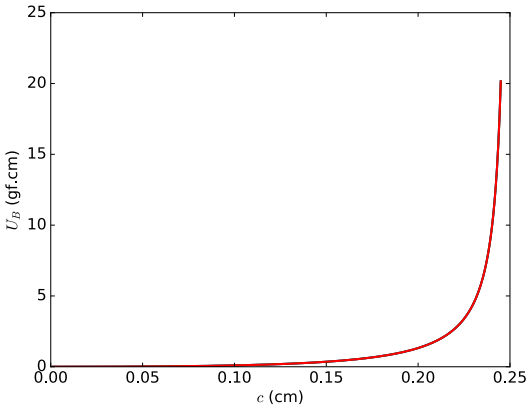
(a) From bottom, $\mu_k = [0.1, 0.3, 0.5]$



(b) Corresponding values of U_{M_F}



(c) Corresponding values of U_{M_S}



(d) Corresponding values of U_B

Figure 4.13: Minimised sphere energy for a range of μ_k.

Under this assumption, the intuitive unit cell is a cube with sides of length $2(r_0 + h + t_0)$, containing a sphere of radius r_0 at its centre and with the remainder of the cube containing the fibrous web.

However, assumption 4.3.1 is insufficient for formulating a fully-functional unit cell. The trouble with any choice of unit cell is that coupling the sphere and web together under compression quickly becomes fraught with complexity. The potential range of movement of the sphere within the unit cell gives rise to lateral compression of the web around the equator and non-uniform compression near the axis. The web in between these regions is prone to shearing. None of the existing models of random fibre masses are designed to cope with compression along more than one axis, and so we must either refine the unit cell or refine our assumptions. We opt for the latter approach, and make the following assumption:

Assumption 4.3.2. *The energy of the web component can be modelled by the van Wyk model of fibrous assemblies.*

The van Wyk model was converted to the energy method in section 3.3.4; by knowing the volume changes of the sphere and the overall unit cell, the energy of a van Wyk web component can be determined from their difference. This is a considerable oversimplification, but necessary to progress.

4.3.1 Basic unit cell

Figure 4.14 shows a cross-section of the proposed unit cell. The web and sphere are compressed together between the same two parallel frictionless plates that previously compressed the sphere alone. We define d to be the distance each plate has moved while compressing the unit cell. The parameter c retains its definition from the sphere model, and we introduce a new parameter $b = d - c$ to be the level of compression of the web component. Thus $t = t_0 - b$ is the thickness of the compressed web above and below the sphere.

To couple the sphere and web components together, we make the following assumptions:

Assumption 4.3.3. *There is no friction between the sphere and the web. More precisely, the web is allowed to flow freely around the sphere.*

This is the most unrealistic of our assumptions, particularly in light of the fact that there will be PLA-PLA bonds between the sphere and web components.

Assumption 4.3.4. *The unit cell does not expand laterally during compression.*

This places a geometric constraint on the lateral expansion of the compressed sphere, namely:

$$\begin{aligned} r' + r'' &\leq r_0 + t_0 \\ r' + (r_0 - c) &\leq r_0 + t_0 \\ r' - c &\leq t_0 \\ r' &\leq c + t_0. \end{aligned} \tag{4.46}$$

This assumption about the boundary conditions makes sense if we consider a full-size knoppy web under uniform compression. The lateral dimensions of the knoppy web are sufficiently larger than the dimensions of the unit cell that we can consider them to be infinite. This places a periodic boundary condition on the side walls of the unit cell that fixes them in place.

Assumption 4.3.5. *At any given level of overall compression d, the sphere and web components are compressed such that the total energy of the system is minimised.*

From this assumption we define $0 \leq q_c \leq 1$ to be the fraction of overall compression that is associated with the sphere. d and q_c together form the variables of the basic knoppy web unit cell. q_c is incorporated into the model through its relationship with c. In the simple case of small compression, where $d << r_0$

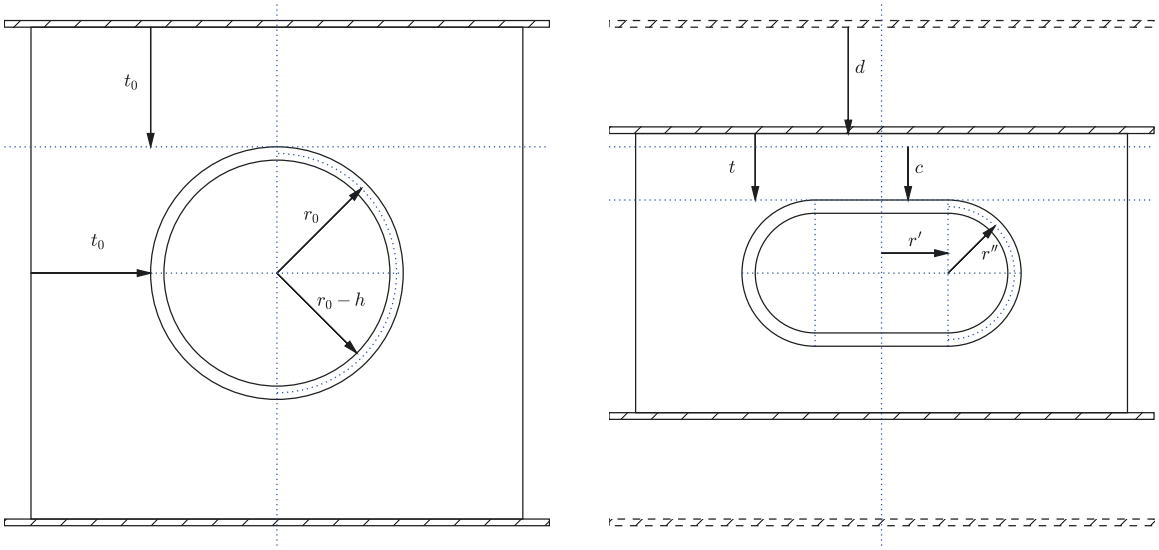


Figure 4.14: Compression of the basic knoppy web unit cell

and $d << t_0$, the compression can be fully associated with either the sphere or web component, and thus $c = q_c d$. However, for large compression this is not necessarily the case—one component may be squashed flat, if the relative stiffnesses are significantly different. In section 4.2.5 we simulated sphere compression as far as was geometrically possible, but gave a soft limit of $c \leq \frac{3}{4}r_0$, or about 75% compression. In later experiments presented in chapter 5, we compress knoppy web by as much as 83%; restricting sphere compression to 75% may therefore incorrectly simulate the contribution of the knops. Therefore, we choose compression of either the sphere or web by 7/8 as the point where steric limitations will inhibit further compression. This condition gives rise to the following relationship between c and q_c:

$$c = \begin{cases} q_c d, & d \leq \frac{7}{8}(r_0 - h) \text{ and } d \leq \frac{7}{8}t_0 \\ q_c d + (1 - q_c)(d - \frac{7}{8}t_0), & d \leq \frac{7}{8}(r_0 - h) \text{ and } d > \frac{7}{8}t_0 \\ q_c \frac{7}{8}(r_0 - h), & d > \frac{7}{8}(r_0 - h) \text{ and } d \leq \frac{7}{8}t_0 \\ q_c \frac{7}{8}(r_0 - h) + (1 - q_c)(d - \frac{7}{8}t_0). & d > \frac{7}{8}(r_0 - h) \text{ and } d > \frac{7}{8}t_0 \end{cases} \tag{4.47}$$

Thus the limit on d is

$$0 \leq d \leq \frac{7}{8}(r_0 - h + t_0). \tag{4.48}$$

Because eq. (4.46) depends on c, we also introduce $0 \leq q_{r'} \leq 1$ to represent the expansion of the sphere as a fraction of the maximum possible expansion; by definition,

$$r' = q_{r'}(c + t_0). \tag{4.49}$$

4.3.1.1 Web volumes

Recall that eq. (3.36) calculates the energy of a van Wyk fibrous assembly from its change in volume. The initial volume v_0 is simply the uncompressed unit cell volume minus the volume occupied by the uncompressed sphere,

$$v_0 = 8(r_0 + h + t_0)^3 - \frac{4}{3}\pi(r_0 + h)^3, \tag{4.50}$$

while the compressed volume v_c is the compressed unit cell volume minus the volume occupied by the compressed sphere,

$$v_c = 8(r_0 + h + t_0)^2(r'' + h + t) - v_{cs}. \tag{4.51}$$

To obtain the compressed volume of the sphere, we leverage the axial symmetry to calculate a volume of rotation. The outer cross-section of the sphere under compression can be described by

$$\rho = r' + \sqrt{(r'' + h)^2 - z^2}, \tag{4.52}$$

where z is along the axis of compression. Thus the volume of the compressed sphere is

$$\begin{aligned} v_{cs} &= \int_{-(r''+h)}^{r''+h} \pi\rho^2 dz \\ &= \pi \int_{-(r''+h)}^{r''+h} \left(r' + \sqrt{(r'' + h)^2 - z^2}\right)^2 \, \mathrm{d}z \\ &= \pi \int_{-(r''+h)}^{r''+h} \left(r'^2 + 2r'\sqrt{(r'' + h)^2 - z^2} + ((r'' + h)^2 - z^2)\right) \, \mathrm{d}z \\ &= \frac{1}{3}\pi(r'' + h)\left(6r'^2 + 3\pi r'(r'' + h) + 4(r'' + h)^2\right). \end{aligned} \tag{4.53}$$

4.3.1.2 Total energy

By the energy method, the energy of the overall knoppy web unit cell is equal to the sum of the energies of the sphere and the web regions. Thus, we can write the total energy as

$$\begin{aligned} U_T &= U_S + U_W \\ &= 2U_{MF} + U_{MS} + U_B + U_{vW}. \end{aligned} \tag{4.54}$$

The basic unit cell model has six geometric parameters:

Table 4.2: Parameters for the basic knoppy web model

Parameter		Units	Number of steps	Value
	d	cm	200	$[0.001, \frac{7}{8}(r_0 - h) + \frac{7}{8}t_0]$
	q_c		200	$[0.001, 0.999]$
	$q_{r'}$		200	$[0.001, 0.999]$
	r_0	cm		0.25
	h	μm		50
	t_0	cm		0.25
	E_k	gf/cm^2		5.1 x 10^5
	μ_k			0.5
Fibre modulus	E_f	gf/cm^2		3.98 x 10^7
Packing density	ρ_w	g/cm^3		0.03
Fibre density	ρ_f	g/cm^3		1.3
van Wyk constant	K			0.001

- d, the amount by which the unit cell has been compressed.
- q_c, the fraction of overall compression that is associated with the knop.
- $q_{r'}$, the lateral expansion of the knop as a fraction of the maximum possible expansion.
- r_0, the radius of the uncompressed knop in spherical polar coordinates.
- $2h$, the thickness of the knop wall.
- t_0, the thickness of the web surrounding the knop.

Additionally, there are four material parameters:

- E_k, the Young's modulus of the knop membrane.
- μ_k, the Poisson's ratio of the knop membrane.
- K, the van Wyk constant of the web.
- ρ_w, the density of the web.

r_0, h, E_k and μ_k are constant properties of the knop, per section 4.2.4. We similarly state that t_0, K and ρ_w are properties of the web, and are considered to be constants throughout the simulation. Thus d, q_c and $q_{r'}$ become the simulation parameters, over which the energy is calculated. In this case, d is the independent variable, and we use the minimum energy principle to define the values of q_c and $q_{r'}$.

4.3.1.3 Analysis

The basic knoppy web unit cell model was implemented in Python 2.7 [82], leveraging the same packages used for the simple sphere model (listed in section 4.2.5). The combination of SymPy and IPython enabled the sharing of sphere model equation code between the two implementations. A full code listing is provided in appendices B.3 and B.4.

Table 4.2 shows the values of the various parameters for the model. r_0, E_k and μ_k reprise their values from table 4.1. t_0 was chosen to be equal to r_0, so that the relative effects of compression on the sphere and web components were more easily rendered. E_f, ρ_w and ρ_f are typical values for New Zealand wool fibres [72]. K is chosen semi-arbitrarily for this behavioural analysis, for reasons discussed later in section 5.5.1.4.

Total energy Figures 4.15a and 4.15b show the surface and contour plots for the model. They both show that the relative compression within the unit cell is heavily biased in favour of the sphere. The minimum energy path plotted in fig. 4.15b shows that there is no sphere compression until the plates are in contact with it.

Figure 4.15d shows the total strain energy in the sphere and web components along the minimum energy path. The web strain energy follows a reasonably smooth curve that is approximately exponential above 20% compression. The sphere strain energy does not start increasing significantly until around $d = 0.22\,\text{cm}$, which as noted above is where the plates start to contact it. Once sphere compression begins, its strain energy climbs much more quickly than the web strain energy. Ideally there would be zero contribution from the sphere below $d = 0.22\,\text{cm}$; in reality, there is some non-zero noise caused by the numerical integration.

Energy components Figure 4.16 shows the energy behaviour of the sphere component. As expected from intuition, the preferred compression path of the sphere component is for the web component to compress first, ie. $c = 0$ until $d > \frac{7}{8}t_0$. There is some instability at very small c, but we believe this to be a numerical integration issue (and the cause of the noise in fig. 4.15d). Beyond that, there is no variation in energy across constant c; slices of fig. 4.16a along $c = d - b$ for constant b will be identical. This reflects the minimal coupling between the sphere and web components from assumption 4.3.3.

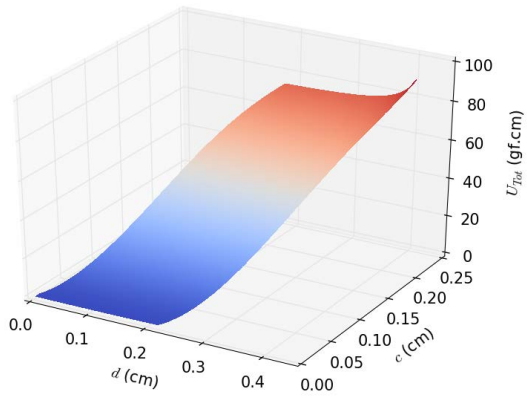
Figure 4.17 shows the energy behaviour of the web component. The preferred compression path of the web component is for the sphere component to compress first, as expected. However, there is an obvious and significant issue with the overall energy behaviour: the contours in fig. 4.17b are close to vertical. In particular, the energy at $(d, c) \approx (0.22\,\text{cm}, 0.22\,\text{cm})$ is very similar to that at $(d, c) \approx (0.22\,\text{cm}, 0\,\text{cm})$. But these are diametrically opposite situations, where either the sphere has completely compressed, or it has not compressed at all. In the former case, the web is compressed only due to the overall decrease in volume of the unit cell, wehereas in the latter case the web is highly compressed above and below the sphere. Thus intuitively there should be some non-trivial energy difference between these two states—the web strain energy should be significantly higher in the latter case. However, the energy surface accurately reflects the basic unit cell model; assumption 4.3.3 is the direct cause of this behaviour. By assuming that the web component has uniform strain and can flow around the sphere, the basic unit cell model cannot account for the presence of high-strain regions.

Effect of parameters There are three parameters in this model: the van Wyk constant K, the web density ρ_w, and the web thickness t_0 (section 4.3.1.2).

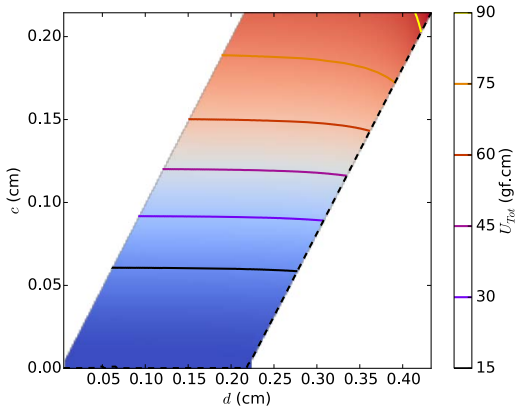
Figure 4.18 shows the effect of varying K on the minimised strain energy. K is a linear scaling factor on U_{Web} per eq. (3.36), and the effect on U_{Tot} can be seen in fig. 4.18b: as the web gets stiffer, the contribution from the sphere becomes less dominant. This is reflected in figs. 4.18c and 4.18d—the U_{Sphere} curve shape is unchanged, while the U_{Web} curve scales up. However, U_{Sphere} and U_{Web} both have show step in the energy, that grows larger and occur sooner as K is increased. The steps are equal and opposite in magnitude, leading to a U_{Tot} that is continuous but not piecewise smooth (fig. 4.18a).

The cause of the step can be seen in fig. 4.19. As K is varied from 0.001 to 0.1, the web becomes stiffer relative to the sphere, and the energy surface visibly transitions from mostly sphere-like behaviour to mostly web-like behaviour. In all cases the minimum energy paths show that at early stages of compression the web component takes up all the strain, followed by the sphere once the compression surfaces reach it. However, eventually a compression level is reached where it is energetically more favourable for the sphere to be completely compressed, and the system changes configuration into the preferred state. This is very similar to the buckling that occurs when a hollow sphere is compressed ((section 3.2.1), and is unrealistic for a knop as we have previously established.

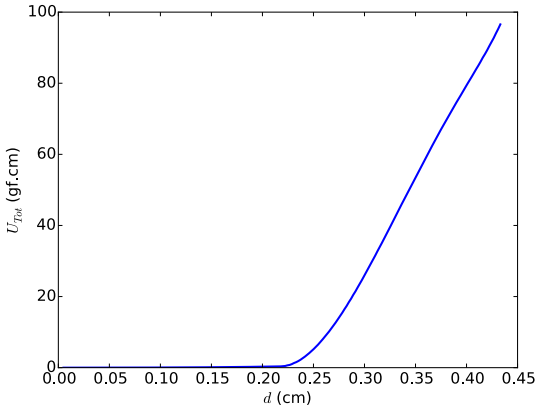
Figure 4.20 shows the effect of varying ρ_w on the minimised strain energy. ρ_w is a cubic scaling factor on U_{Web} per eq. (3.36), and therefore has approximately the same effect as K but on a different scale. It



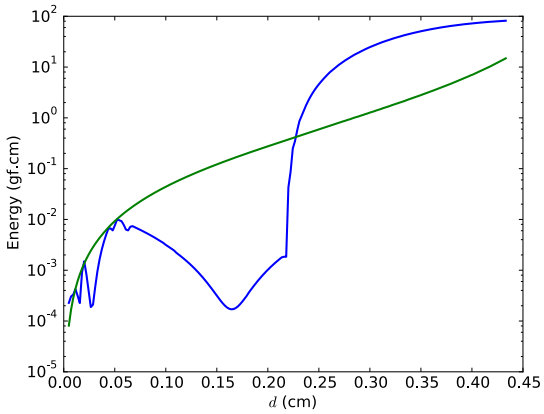
(a) Energy surface, minimised along r'.



(b) Corresponding energy contour, with dotted line showing minimum energy path.

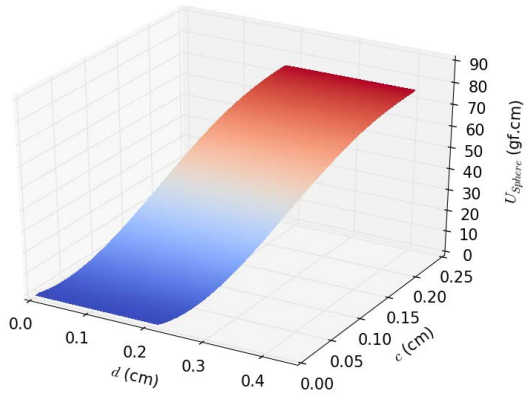


(c) Minimum energy curve

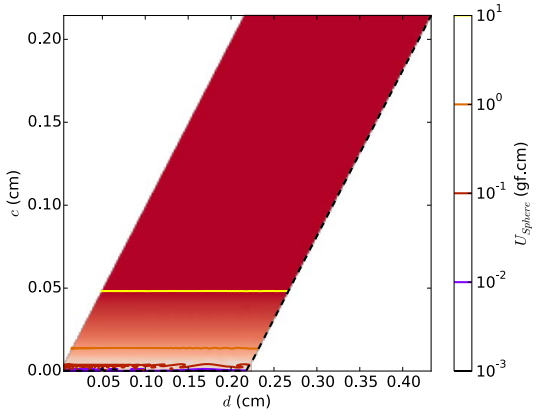


(d) Sphere and web contributions to the minimum energy. The blue line is the sphere component, and the green line is the web component. Below $d = 0.22\,\text{cm}$, the sphere component is noise.

Figure 4.15: Basic knoppy web model

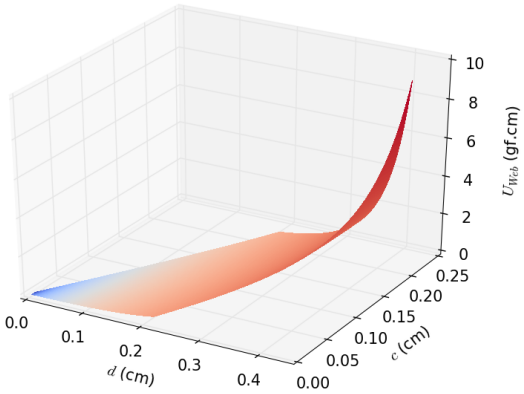


(a) Energy surface

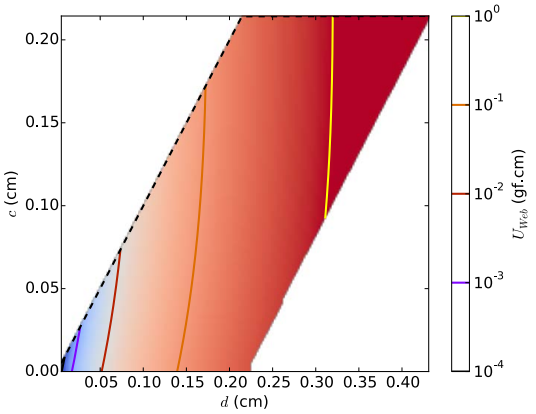


(b) Corresponding energy contour, with dotted line showing minimum energy path.

Figure 4.16: Strain energy within the sphere, minimised along r'.

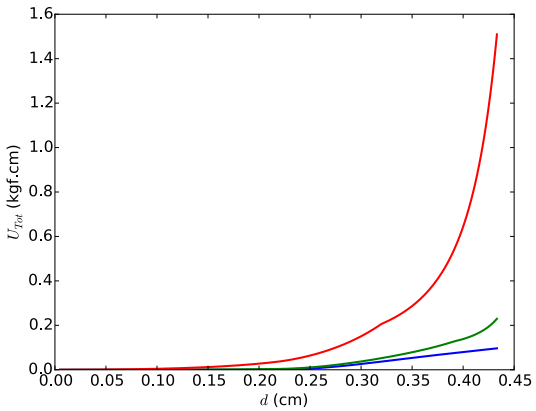


(a) Energy surface

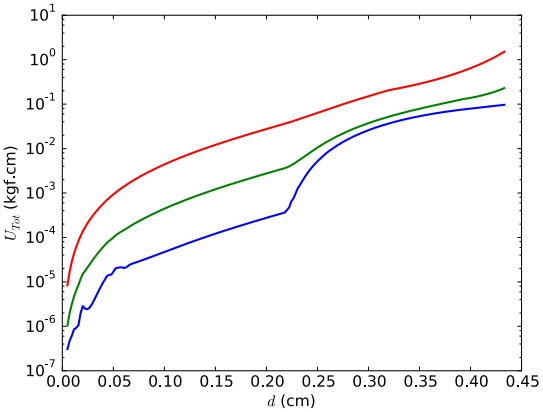


(b) Corresponding energy contour, with dotted line showing minimum energy path.

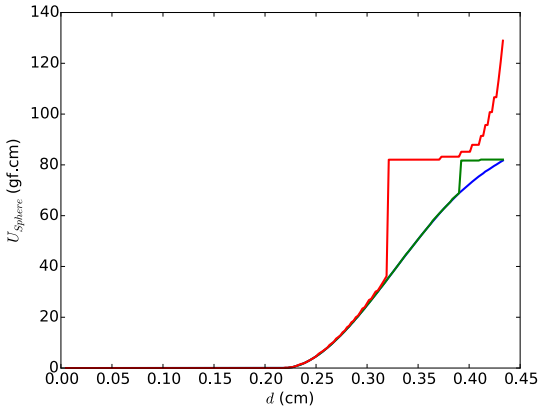
Figure 4.17: Strain energy within the web, minimised along r'.



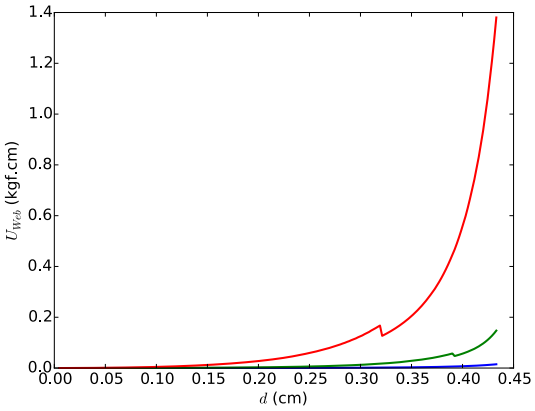
(a) From bottom, $K = [0.001, 0.01, 0.1]$



(b) Corresponding log plot

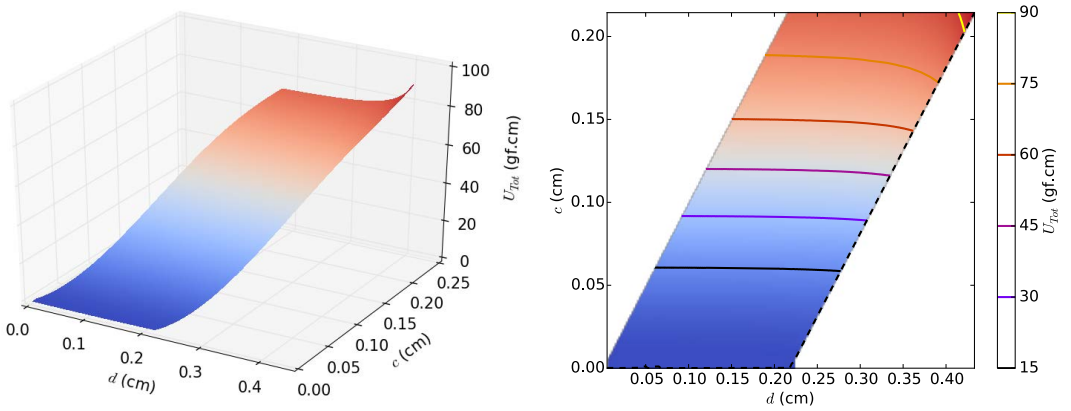


(c) Corresponding values of U_{Sphere}

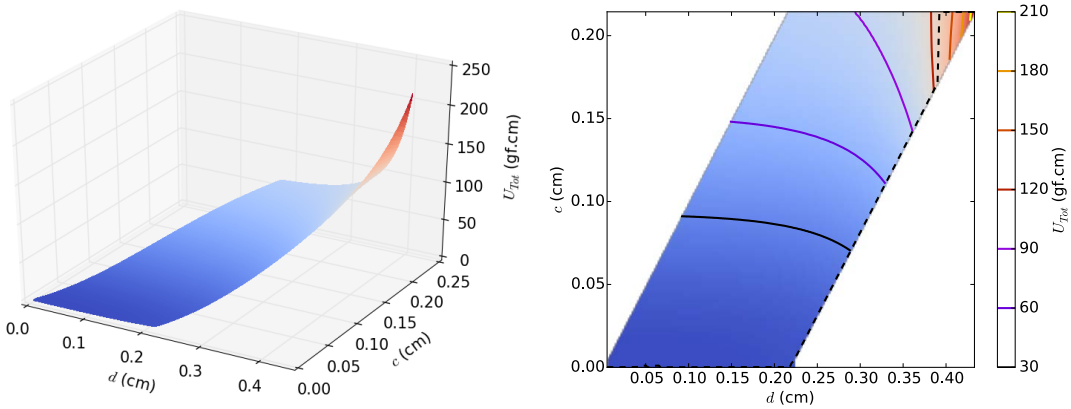


(d) Corresponding values of U_{Web}

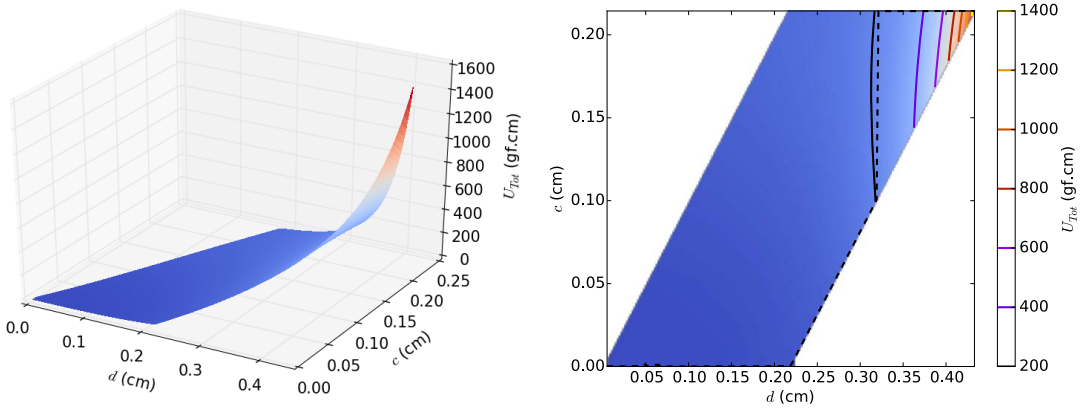
Figure 4.18: Minimised basic unit cell energy for a range of K.



(a) $K = 0.001$

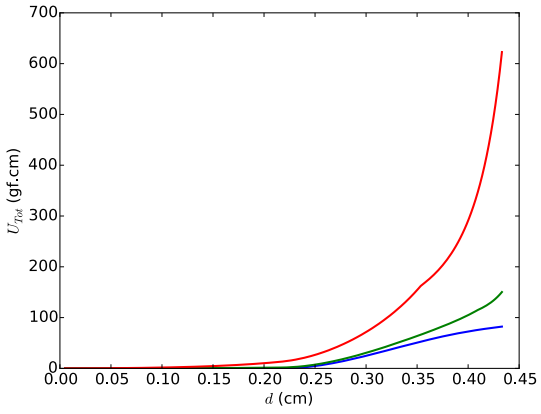


(b) $K = 0.010$

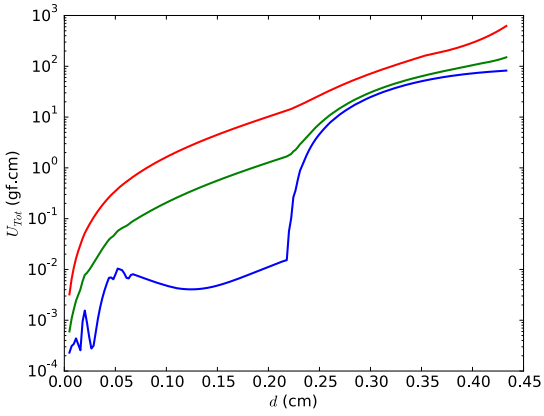


(c) $K = 0.100$

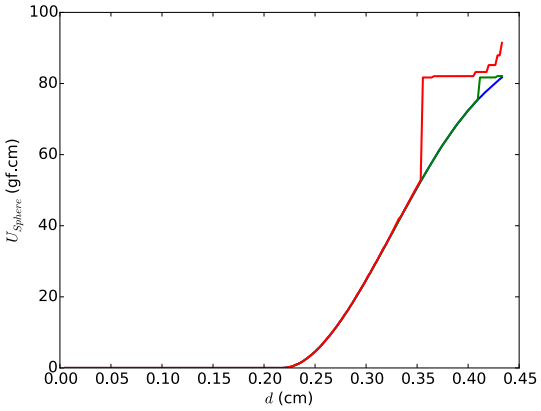
Figure 4.19: Basic unit cell energy surfaces for a range of K, minimised along r'.



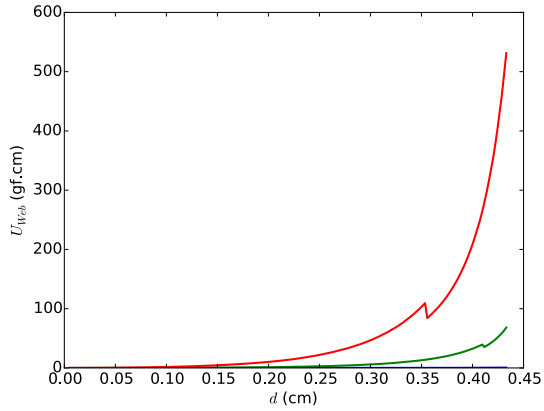
(a) From bottom, $\rho_w = [0.01, 0.05, 0.1]\mathrm{g\,cm^{-3}}$



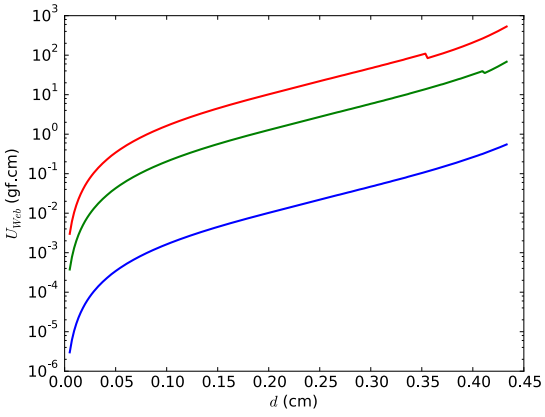
(b) Corresponding log plot



(c) Corresponding values of U_{Sphere}



(d) Corresponding values of U_{Web}



(e) Log plot of U_{Web}

Figure 4.20: Minimised basic unit cell energy for a range of ρ_w.

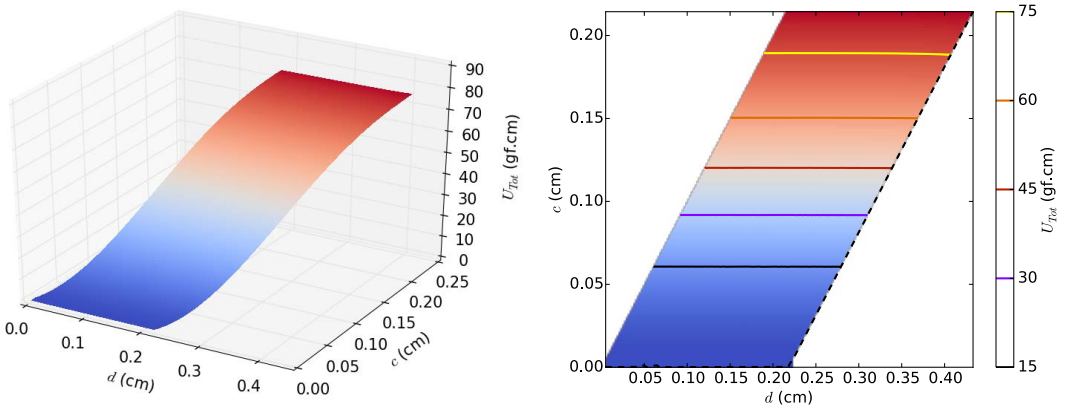
also experiences the same geometric discontinuity, as visible in figs. 4.20c, 4.20d and 4.21.

Figure 4.22 shows the effect of varying t_0 on the minimised strain energy. t_0 defines the volume of the web component, as well as the volume of the overall unit cell; thus it has a more complex relationship with strain energy than K and ρ_w. Because d has an absolute geometric upper limit of $r_0 + h + t_0$, increasing t_0 increases the range of compression (which can be seen on both the minimised plots in fig. 4.22 and the energy surfaces in fig. 4.23).

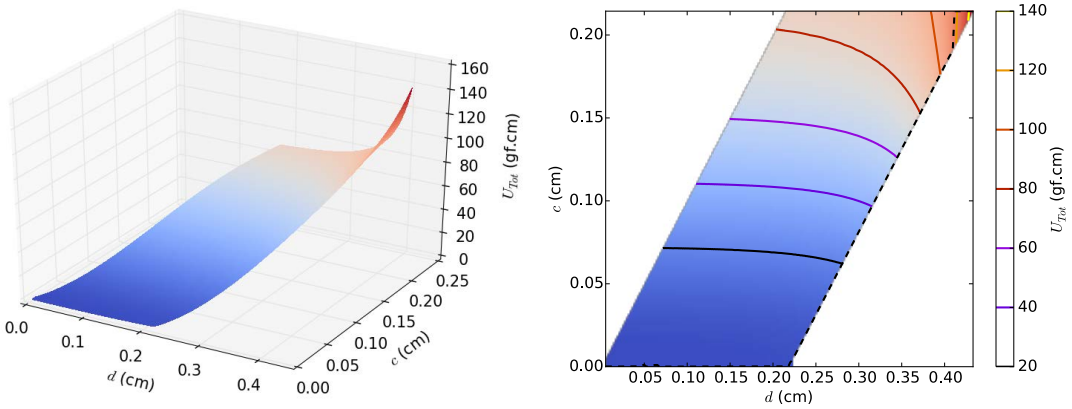
The value of K for these simulations is at the low end of the range shown in fig. 4.18, while the value of t_0 used there is at the high range of these figures. Thus U_{Web} is very low compared to U_{Sphere}, and fig. 4.22a is very similar to fig. 4.22c except for at very high compression.

Intuitively, increasing t_0 should increase the strain energy stored in the web; this is observed in fig. 4.22d. The sphere component has no direct dependency on t_0, and thus there is in general no change to the U_{Sphere} energy curve (fig. 4.22c), aside from a shift in the onset of strain energy caused by the sphere being further from the compression plates due to the thicker web component.

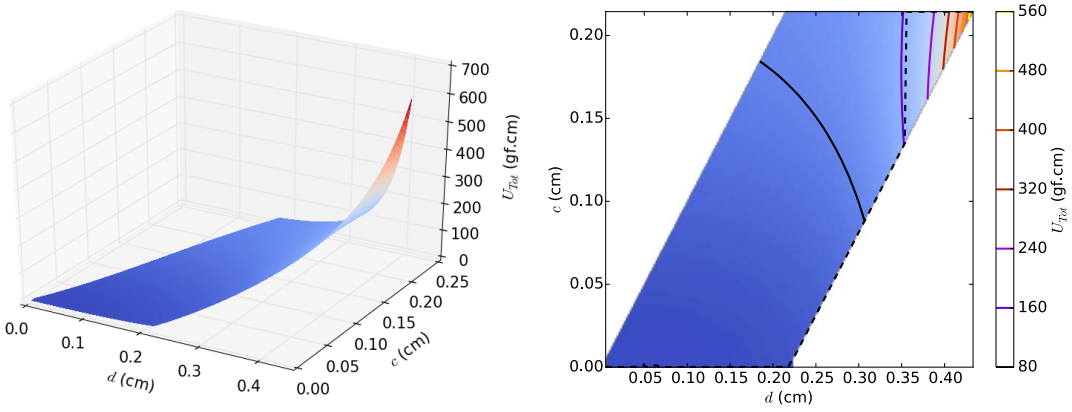
One notable difference between the U_{Sphere} energy curves in fig. 4.22c is that for $t_0 = 0.05\,\text{cm}$, there is a sudden upturn in energy at about $d = 0.17\,\text{cm}$. The cause of this is assumption 4.3.4: the unit cell does not undergo lateral expansion, and at this small t_0 the sphere's most energetically favourable lateral expansion would take it past the boundaries of the unit cell. That is, once the upper constrain on r' given by eq. (4.46) is reached, the sphere is forced to maintain a higher-energy configuration than its ideal minimum.



(a) $\rho_w = 0.01\,\mathrm{g\,cm^{-3}}$

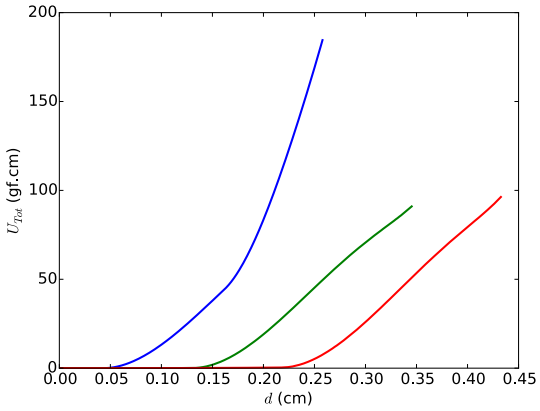


(b) $\rho_w = 0.05\,\mathrm{g\,cm^{-3}}$

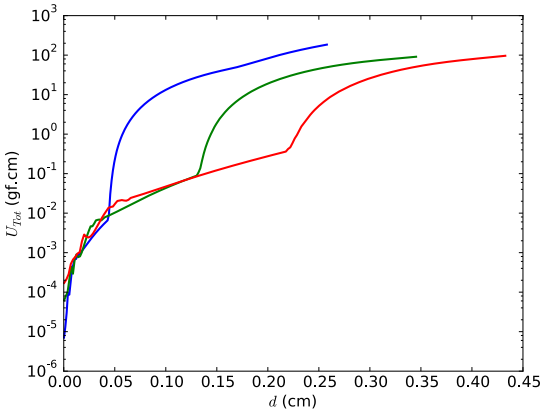


(c) $\rho_w = 0.10\,\mathrm{g\,cm^{-3}}$

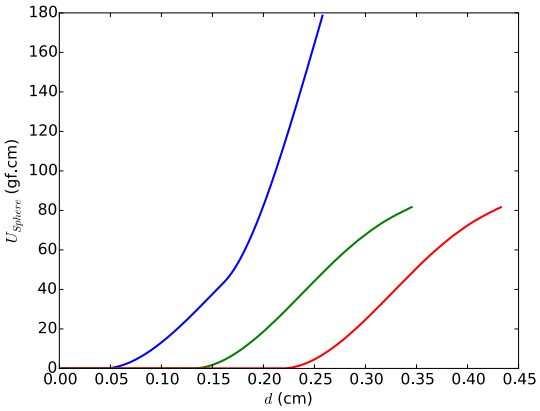
Figure 4.21: Basic unit cell energy surfaces for a range of ρ_w, minimised along r'.



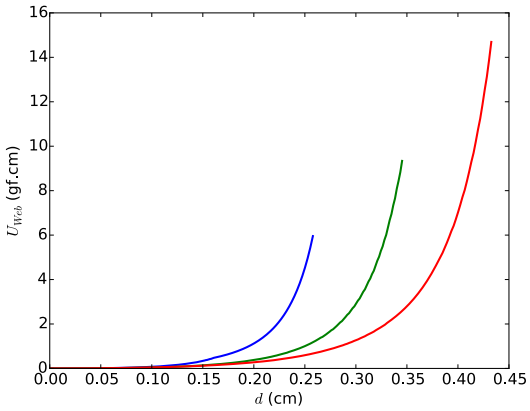
(a) From left, $t_0 = [0.05, 0.15, 0.25]$cm



(b) Corresponding log plot

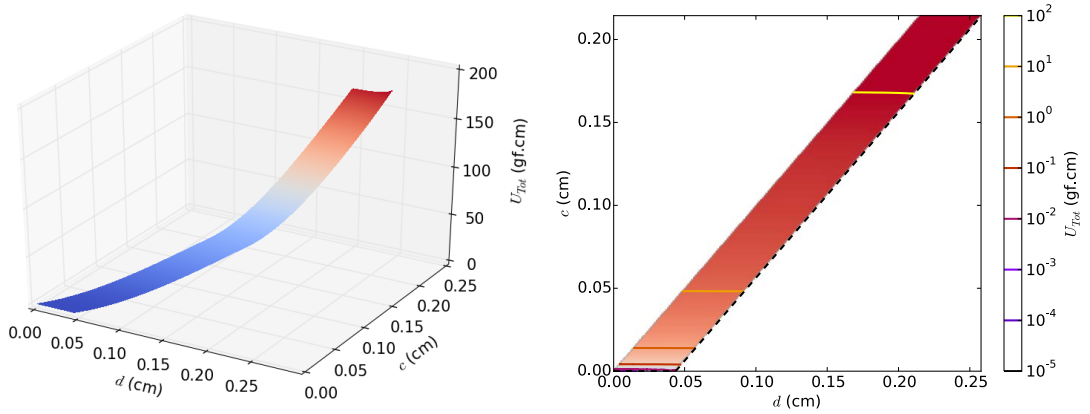


(c) Corresponding values of U_{Sphere}

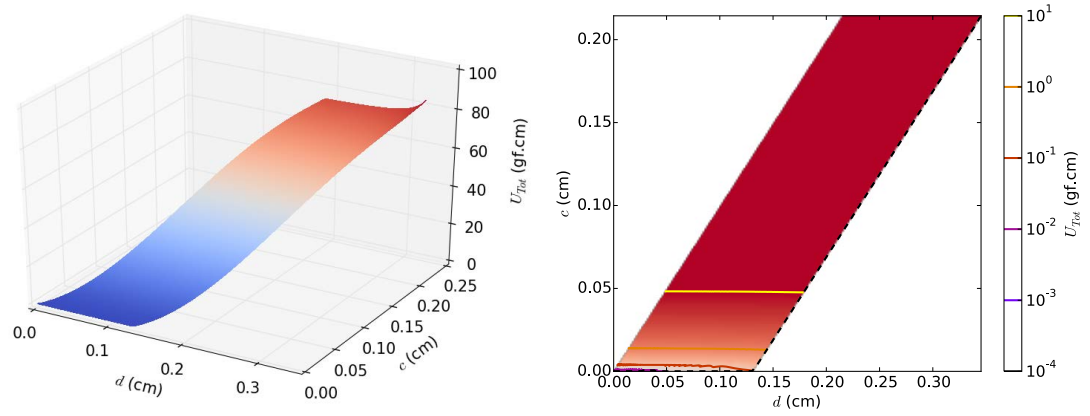


(d) Corresponding values of U_{Web}

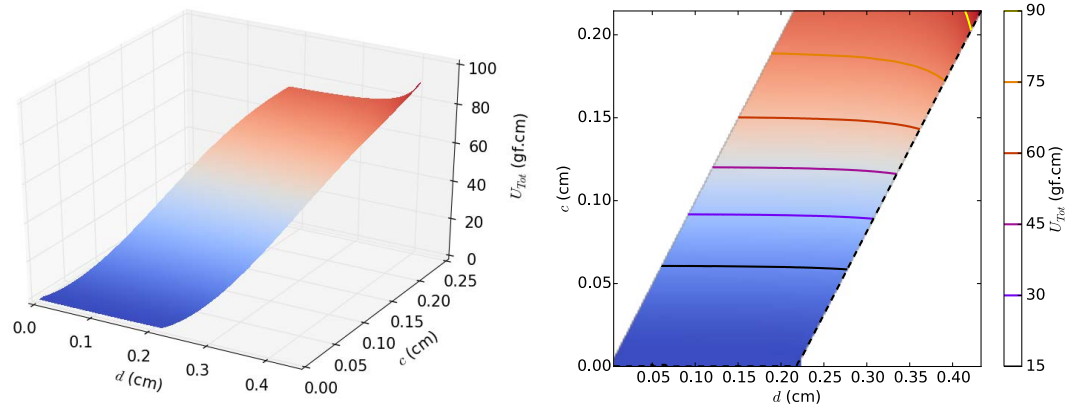
Figure 4.22: Minimised basic unit cell energy for a range of t_0.



(a) $t_0 = 0.05\,\mathrm{cm}$



(b) $t_0 = 0.15\,\mathrm{cm}$



(c) $t_0 = 0.25\,\mathrm{cm}$

Figure 4.23: Basic unit cell energy surfaces for a range of t_0, minimised along r'.

4.3.2 Modified unit cell

The simplest possible form of the knoppy web unit cell is fundamentally inconsistent with the underlying physics. The cause is the assumption that the web fibre "flows" around the knop; physically this means that the web fibre exerts a uniform pressure over the surface of the sphere, instead of a compressive force along its axis.

Part of the problem with the basic unit cell is that assumption 4.3.3 implies that the web fibre has a uniform strain energy throughout. This simplifies the calculation, but we intuitively expect that the web fibre above and below the knop will develop larger strain energies than the off-axis fibres. In this section we propose a modified version of the previous unit cell model that addresses these issues, based on the following assumption:

Assumption 4.3.6. *The web component can be treated as three regions around the knop with identical properties.*

Figure 4.24 shows a cross-section of the proposed modified unit cell. The general layout of the unit cell is unchanged—a single sphere is placed within a cube of web, and the two components are compressed together between two parallel frictionless plates. However, we now partition the web component into three regions: the regions above and below the knop, and the region around it (in the same plane). Each of these three regions is functionally the same as the web component in the basic model—they are treated as van Wyk fibrous masses. But the difference from the basic model is that the web fibres cannot move outside their defined regions. More importantly, we make the following assumption:

Assumption 4.3.7. *The boundaries between these regions do not deform under compression.*

It is this new assumption that leads to a more realistic physical model; without it, the boundaries would deform such that this model would behave exactly as the previous model. Assumption 4.3.3 is now only applied to the middle section.

Compared to the previous model, the new model fundamentally alters the way that pressure is applied to the sphere. In the basic model, the fact that there was a uniform axial pressure applied evenly to the surface of the sphere meant that there was no physical incentive for the sphere to compress until it touched the plates. As the web was compressed the pressure on the sphere increased, but the same pressure that acted to compress the sphere also acted to support it.

Functionally, the new model acts on the sphere in two ways:

- The middle web segment applies pressure to the surface of the outer region of the sphere. It behaves in the same manner as the web component of the basic knoppy web did once the sphere started to compress.
- The web segments above and below the sphere apply axial pressure to the inner region of the sphere (as well as to the middle web segment). These segments model the higher-strain fibrous regions above and below the knop.

It is this split that changes physically the direction of applied pressure: the outer web segments solely apply pressure in the direction of compression, while the inner web segment only applies pressure on the sphere laterally to oppose its radial expansion. Because there is a restriction on the redistribution of web, these forces no longer always balance, leading to more realistic compression behaviour.

4.3.2.1 Web volumes

The regions of web above and below the sphere (the outer web) are identical by symmetry. The initial volume is

$$v_{0,o} = 4t_0(r_0 + h + t_0)^2, \tag{4.55}$$

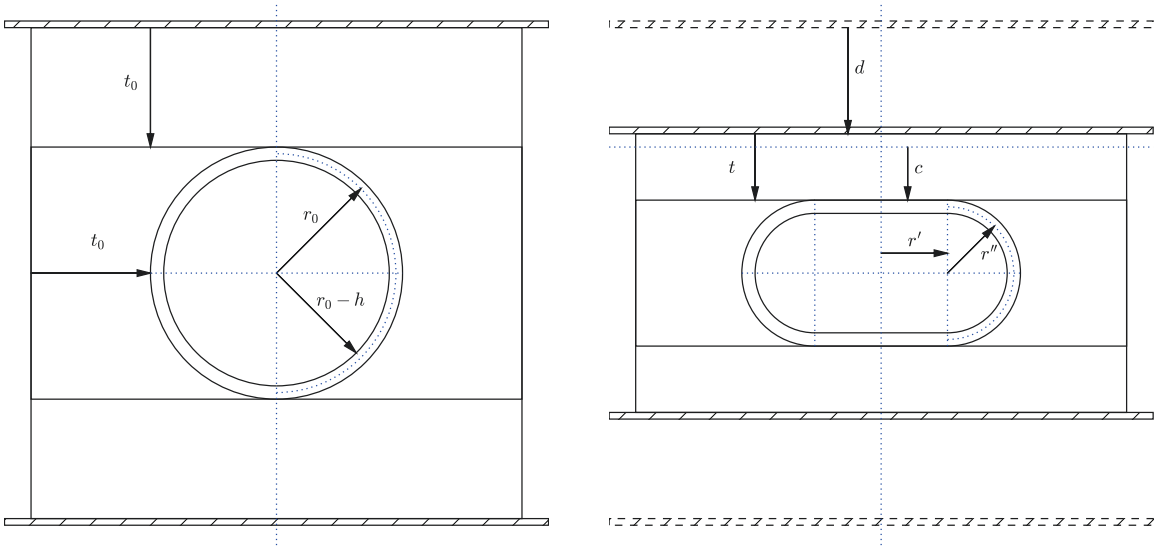


Figure 4.24: Compression of the modified knoppy web unit cell

and the compressed volume is

$$v_{c,o} = 4t(r_0 + h + t_0)^2. \tag{4.56}$$

The region of web around the sphere (the inner web) takes up the remainder of the space in the unit cell. Its initial volume is

$$v_{0,i} = 8(r_0 + h + t_0)^2(r_0 + h) - \frac{4}{3}\pi(r_0 + h)^3, \tag{4.57}$$

and its compressed volume is

$$v_{c,i} = 8(r_0 + h + t_0)^2(r'' + h) - v_{cs}, \tag{4.58}$$

where v_{cs} is given in eq. (4.53).

4.3.2.2 Total energy

By the energy method, the energy of the modified knoppy web unit cell is equal to the sum of the energies of the sphere and the web regions. Thus, we can write the total energy as

$$U_T = 2U_{MF} + U_{MS} + U_B + 2U_o + U_i, \tag{4.59}$$

where

$$U_o = KE_f \left(\frac{\rho_w}{\rho_f}\right)^3 \left[\frac{v_{0,o}^3}{2v_{c,o}^2} + v_{c,o} - \frac{3v_{0,o}}{2}\right] \tag{4.60}$$

is the strain energy within each of the outer web sections, and

$$U_i = KE_f \left(\frac{\rho_w}{\rho_f}\right)^3 \left[\frac{v_{0,i}^3}{2v_{c,i}^2} + v_{c,i} - \frac{3v_{0,i}}{2}\right] \tag{4.61}$$

is the strain energy within the inner web section.

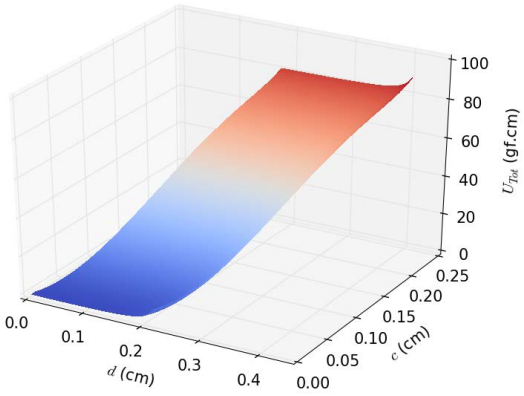
By assumption 4.3.6 the three web regions are modeled using the same material properties. Thus, the parameters of the modified unit cell model are identical to those of the basic unit cell model (defined in section 4.3.1.2).

4.3.2.3 Analysis

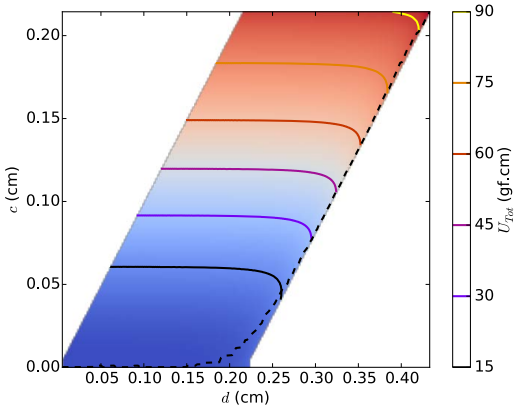
The modified knoppy web unit cell was implemented in the same way as described in section 4.3.1.3. The parameters previously specified in table 4.2 were reused for this model. A full code listing is provided in appendices B.5 and B.6.

Total energy Figures 4.25a and 4.25b show the surface and contour plots for the model. The results of the modification to the unit cell are immediately apparent: under the same chosen parameters as before, the sphere is now beginning to compress before the compression plates are touching it. The minimum energy path given in fig. 4.25b shows that the transition from web-only compression to sphere-only compression is far smoother than previously in fig. 4.15b. This tells us that the new assumption (that the boundary between outer and inner web sections does not deform) is causing the outer web sections to develop much more strain energy than the web component of the basic unit cell did.

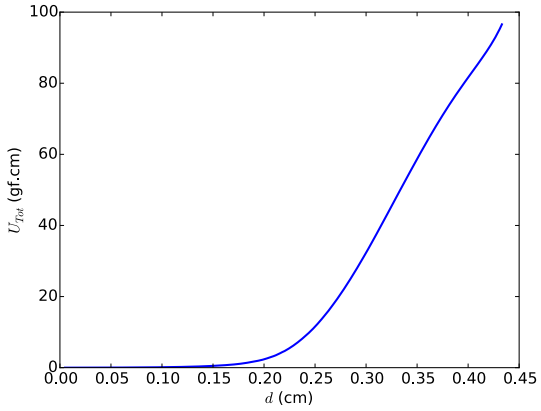
Figure 4.25c shows the strain energy along the minimum energy path as a direct function of d. The curve is approximately the same shape as previously in fig. 4.15c, but with an earlier mid-compression strain onset (as discussed above), and also slightly more variation at higher compression levels. The variation is caused by the splitting of the web component into two independent regions, per assumptions 4.3.3, 4.3.6 and 4.3.7. This can be seen more clearly by comparing U_{Web} in fig. 4.25d to fig. 4.15d. With the modified unit cell, the web strain energy flattens off once the sphere begins to compress, only to upturn again at high strain. The early-stage contributions come from the outer web regions; the late-stage contribution comes from compression of the inner web region, which does not begin until the sphere starts to compress.



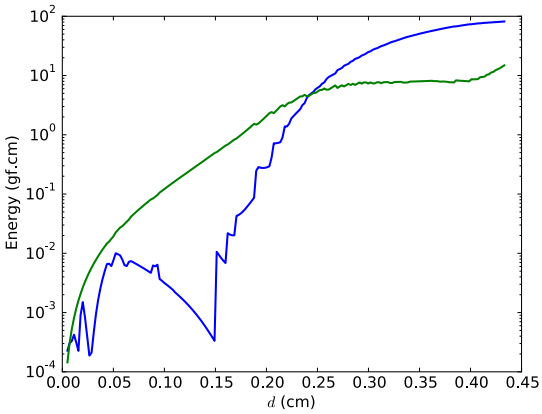
(a) Energy surface, minimised along r'.



(b) Corresponding energy contour, with dotted line showing minimum energy path.

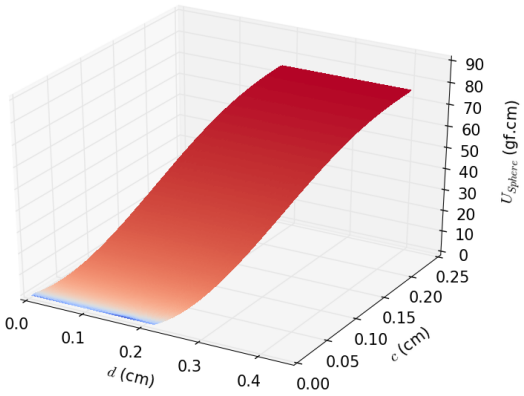


(c) Minimum energy curve

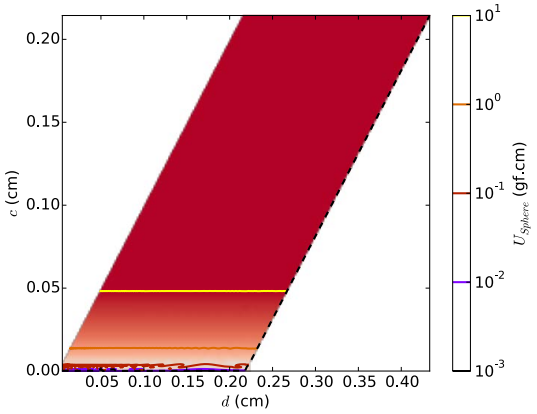


(d) Sphere and web contributions to the minimum energy. The blue line is the sphere component, and the green line is the web component. Below $d = 0.15\,\mathrm{cm}$, the sphere component is noise.

Figure 4.25: Modified knoppy web model



(a) Energy surface



(b) Corresponding energy contour, with dotted line showing minimum energy path.

Figure 4.26: Strain energy within the sphere, minimised along r'.

Energy components Figure 4.26 shows the energy behaviour of the sphere component. It is unaffected by the modifications to the web component, and is therefore identical to fig. 4.16.

Figure 4.27 shows the energy behaviour of the web component; it is distinctly different to the behaviour for the basic unit cell (fig. 4.17). The minimum energy path given in fig. 4.27b shows that for these parameters, the web component favours a roughly equal split between sphere and web compression. This makes sense, because $q_c \to 0$ leads to the outer sections compressing completely before the inner section compresses, while $q_c \to 1$ results in the opposite. By eq. (4.55), the combined initial volume of the outer web regions is

$$2v_{0,o} = 0.605\,\mathrm{cm}^3, \tag{4.62}$$

while by eq. (4.57) the initial volume of the inner web region is

$$v_{0,i} = 0.613\,\mathrm{cm}^3. \tag{4.63}$$

As they have the same inherent properties (by assumption 4.3.6), the inner web region will build up less strain energy than the outer regions for the same change in d. Thus the minimum energy path for U_{Web} will involve slightly more compression of the inner region, as observed in fig. 4.27b.

Effect of parameters There are three parameters in this model: the van Wyk constant K, the web density ρ_w, and the web thickness t_0 (section 4.3.2.2).

Figure 4.28 shows the effect of varying K on the minimised strain energy. U_{Tot} follows the same general trend as for the basic unit cell model, but the energy curves are significantly smoother. Figure 4.28c shows that as K increases and the web component gets stiffer, the sphere compression onsets progressively earlier. This is a significant improvement over the previous model—there are no steps in the energy, and the unit cell is behaving more in line with intuition.

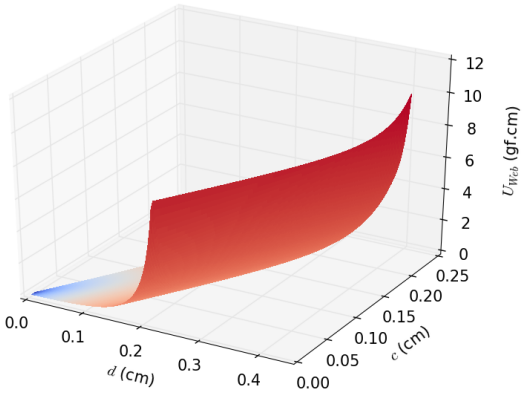
The cause of the improvements can be observed clearly in fig. 4.29. As before, varying K varies the relative contributions of the sphere and web components. But where previously a configuration shift became apparent with higher K, here the minimum energy path smoothly shifts towards earlier sphere compression.

Figure 4.30 shows the effect of varying ρ_w on the minimised strain energy. As with K, the energy curves are considerably smoother, and the sphere compression onsets earlier. The evolution of the minimum energy path shown in fig. 4.31 is very similar to that for K in fig. 4.29.

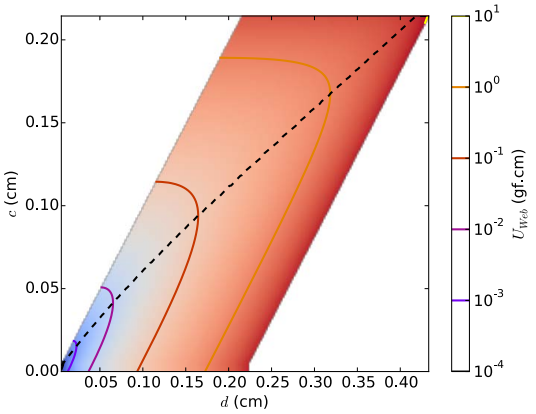
Figure 4.30e is particularly interesting, because here the effect of ρ_w on the individual web regions can be clearly distinguished. By assumption 4.3.6 both web regions have the same stiffness.

- For small ρ_w, the strain energy developed in the outer region is very small compared to the sphere, and so the outer web region will collapse completely before the sphere begins to compress. Then a similar strain buildup occurs in the inner web component as the sphere compresses, but as it is similarly very small, the effect on U_{Web} is almost negligible for most of the remaining compression. Thus U_{Web} is not piecewise smooth, and has two distinct energy stages. These stages are visible in the total energy curve (fig. 4.30b), although it is much smoother due to the sphere contribution.

- For large ρ_w, the stiffness of the web is comparable to (or greater than) the sphere, and so the inner region will start to be compressed sooner. This essentially "blends" the individual energy compression curves of the two regions together, with U_{Web} progressively beginning to approximate a single web region.

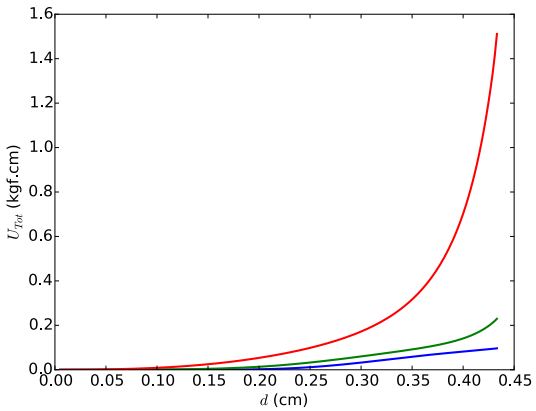


(a) Energy surface

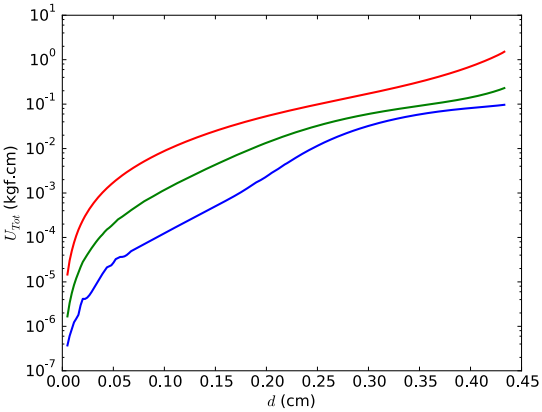


(b) Corresponding energy contour, with dotted line showing minimum energy path.

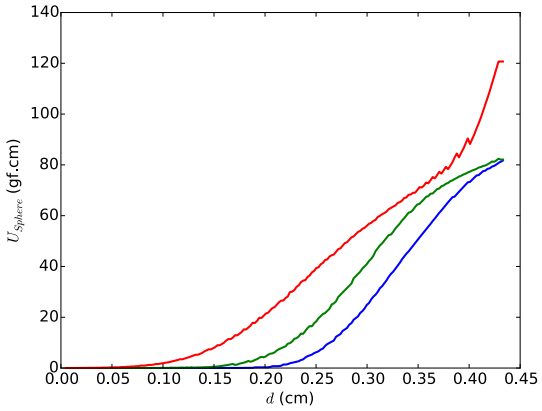
Figure 4.27: Strain energy within the web, minimised along r'.



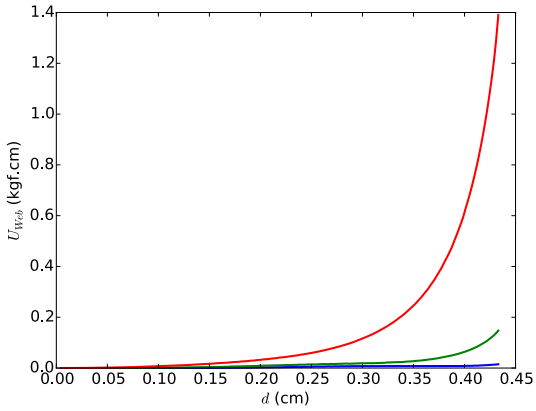
(a) From bottom, $K = [0.001, 0.01, 0.1]$



(b) Corresponding log plot

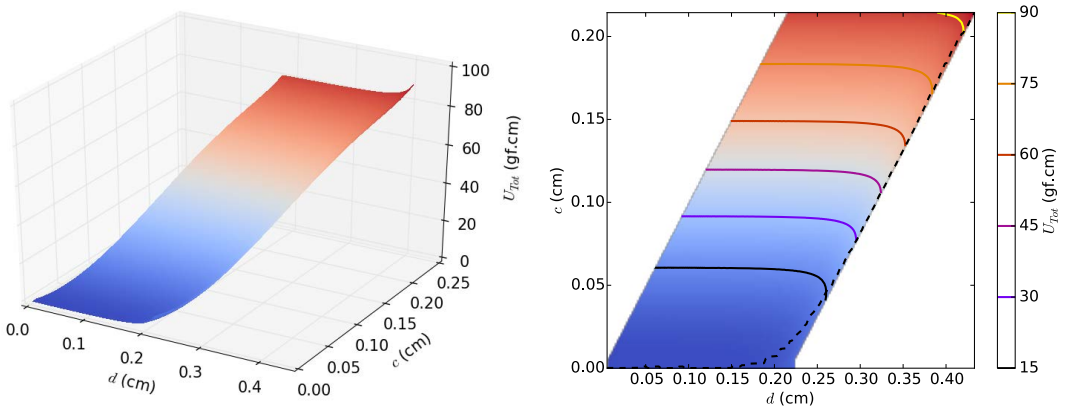


(c) Corresponding values of U_{Sphere}

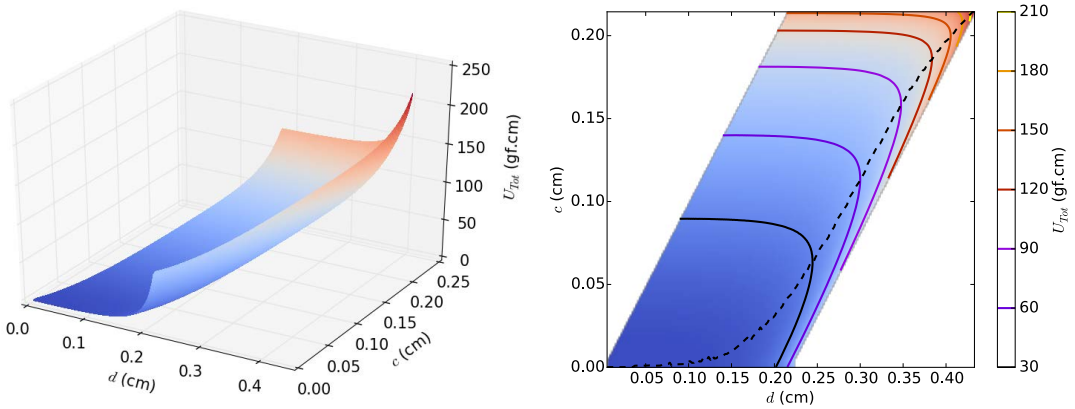


(d) Corresponding values of U_{Web}

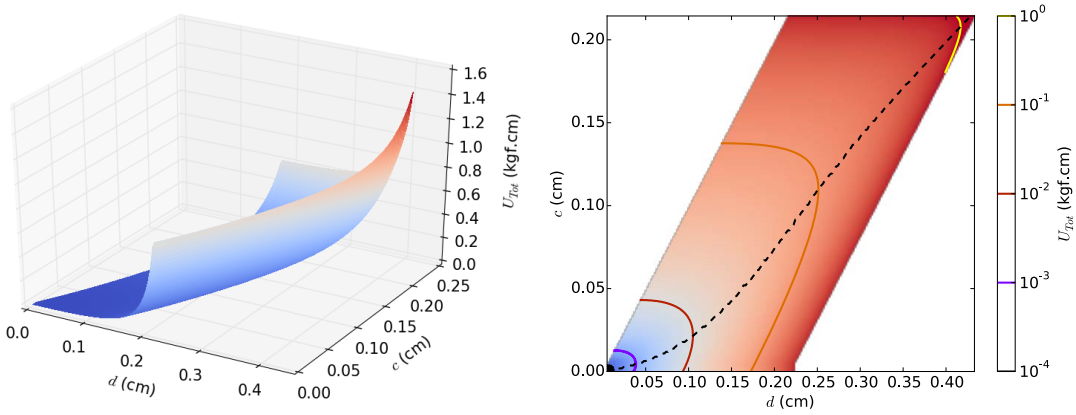
Figure 4.28: Minimised modified unit cell energy for a range of K.



(a) $K = 0.001$

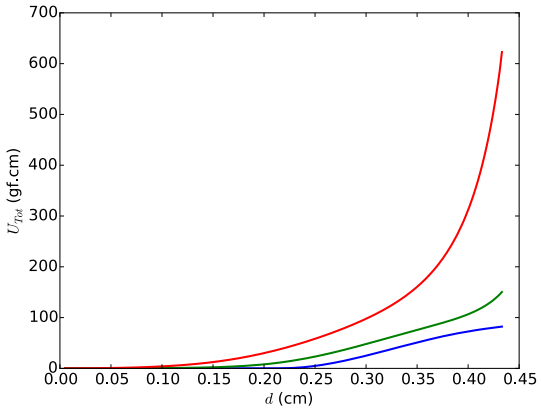


(b) $K = 0.010$

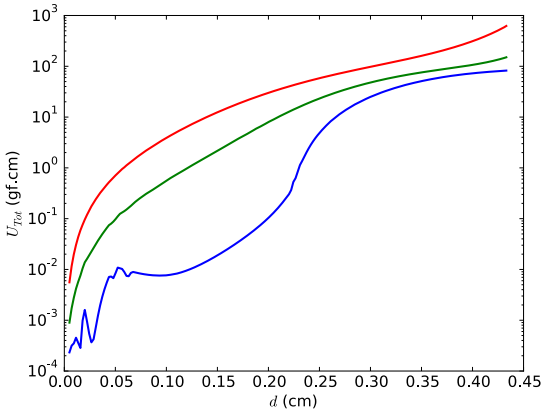


(c) $K = 0.100$

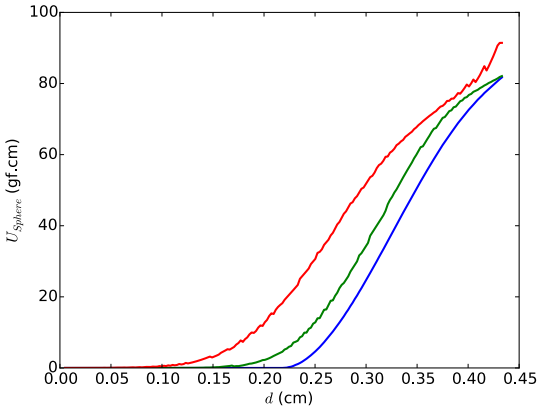
Figure 4.29: Modified unit cell energy surfaces for a range of K, minimised along r'.



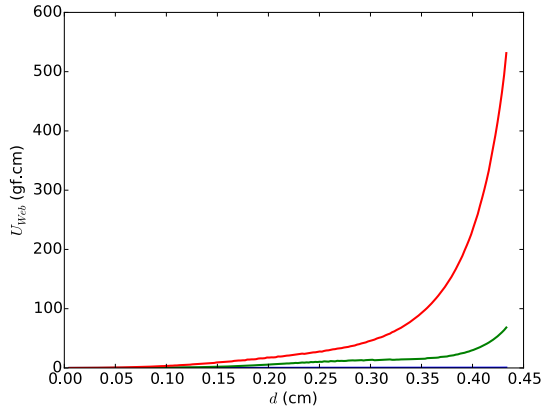
(a) From bottom, $\rho_w = [0.01, 0.05, 0.1]\mathrm{g\,cm^{-3}}$



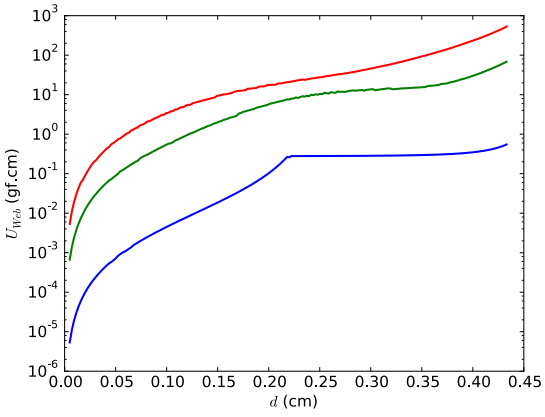
(b) Corresponding log plot



(c) Corresponding values of U_{Sphere}

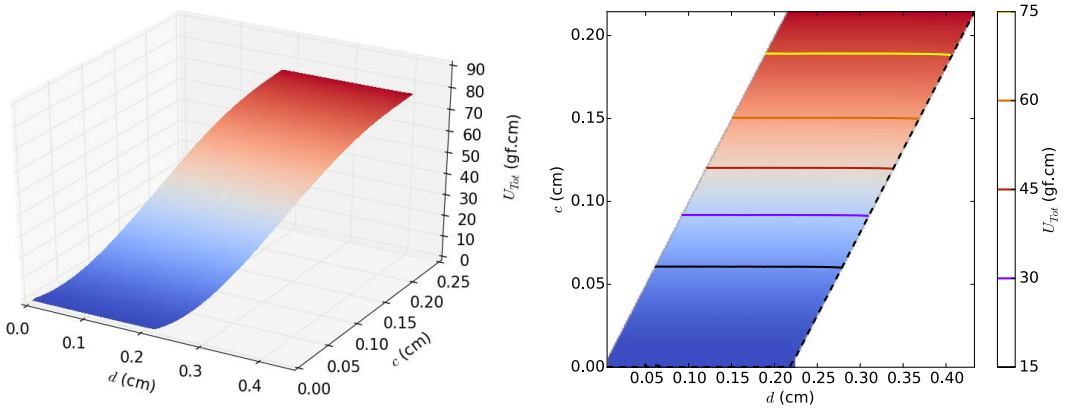


(d) Corresponding values of U_{Web}

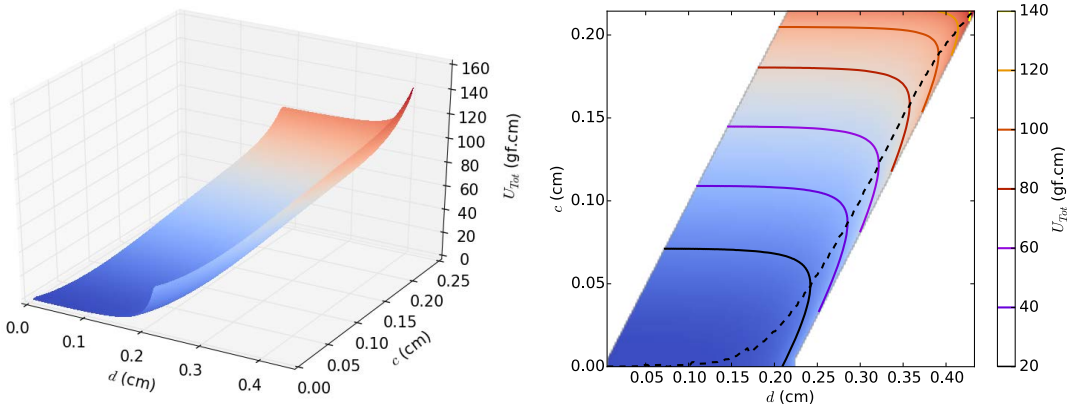


(e) Log plot of U_{Web}

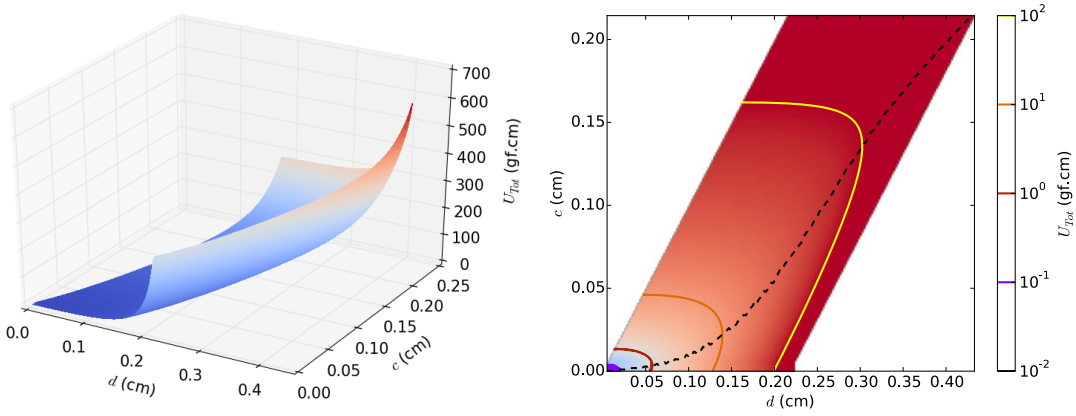
Figure 4.30: Minimised modified unit cell energy for a range of ρ_w.



(a) $\rho_w = 0.01\,\mathrm{g\,cm^{-3}}$



(b) $\rho_w = 0.05\,\mathrm{g\,cm^{-3}}$

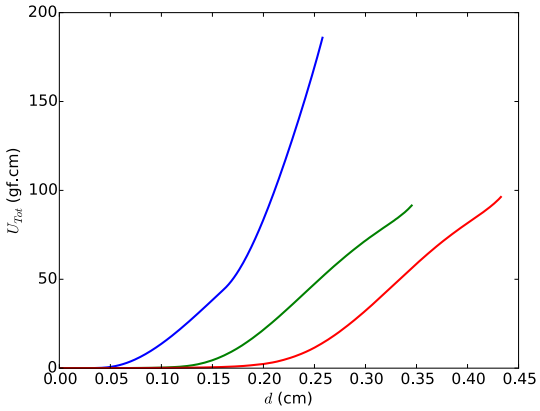


(c) $\rho_w = 0.10\,\mathrm{g\,cm^{-3}}$

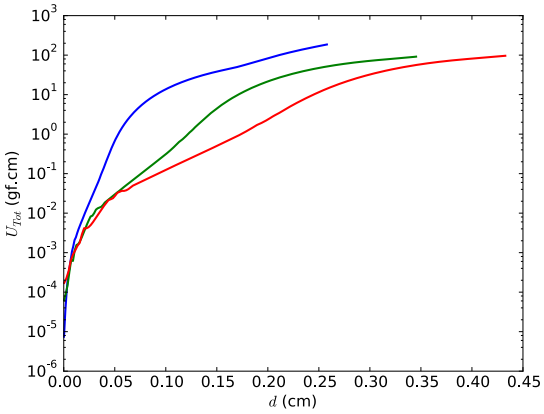
Figure 4.31: Modified unit cell energy surfaces for a range of ρ_w, minimised along r'.

Figure 4.32 shows the effect of varying t_0 on the minimised strain energy. As with the basic unit cell model, the sphere strain energy is much larger than the web strain energy, and thus fig. 4.32a is very similar to fig. 4.32c on a macroscale. Figure 4.32c itself is nearly identical to the previous model, because the sphere is not directly dependent on t_0, and the web strain is insufficient in this case to affect the sphere's lateral spread. The previously-mentioned smoother transition from web compression to sphere compression is visible in both the minimum energy curves (fig. 4.32b lacks the sudden gradient changes present in fig. 4.22b) and the energy surfaces (the minimum energy paths in fig. 4.33 are continuously smooth at low-c medium-d, whereas fig. 4.23 tracked the simulation boundaries).

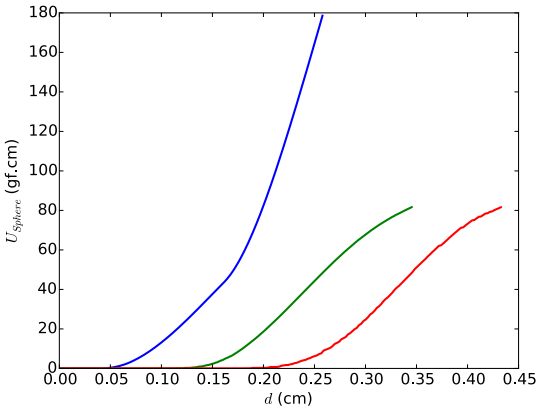
The key difference between the previous model and this one is in the change in shape of the U_{Web} energy curves. The contributions from the two web regions can be clearly observed in fig. 4.32d (due to the low stiffness of the web component, as discussed above for ρ_w). The magnitude of the first phase of the energy curve clearly scales with t_0, corresponding to the combined intial volume of the outer web regions being larger. However, there is almost no visible variance in the magnitude of the second phase. This is because the volume of the inner region is more stable. In addition to the web extending a thickness t_0 away from the sphere, the inner region also includes the web content that fills in the corner regions around the sphere. This web content is present even when $t_0 = 0$, at which point it makes up about 47% of the unit cell volume (see eq. (5.6) in section 5.5.1.3 below). Thus a small increase in t_0 has a much larger relative effect on $v_{0,o}$ than $v_{0,i}$.



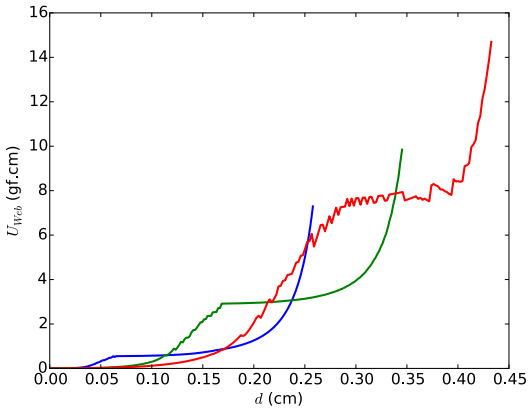
(a) From left, $t_0 = [0.05, 0.15, 0.25]$cm



(b) Corresponding log plot

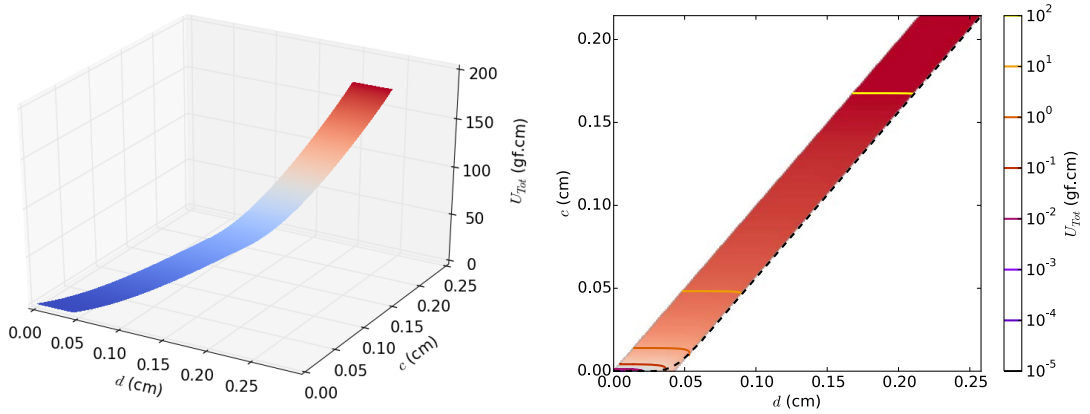


(c) Corresponding values of U_{Sphere}

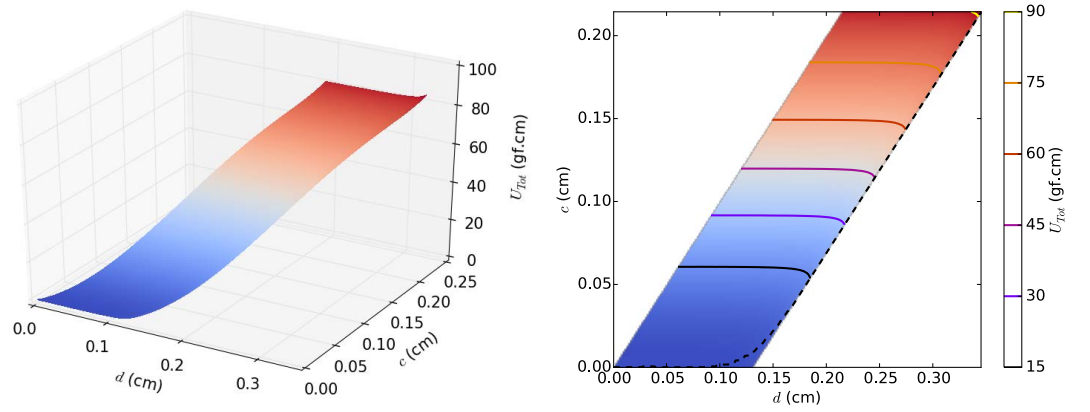


(d) Corresponding values of U_{Web}

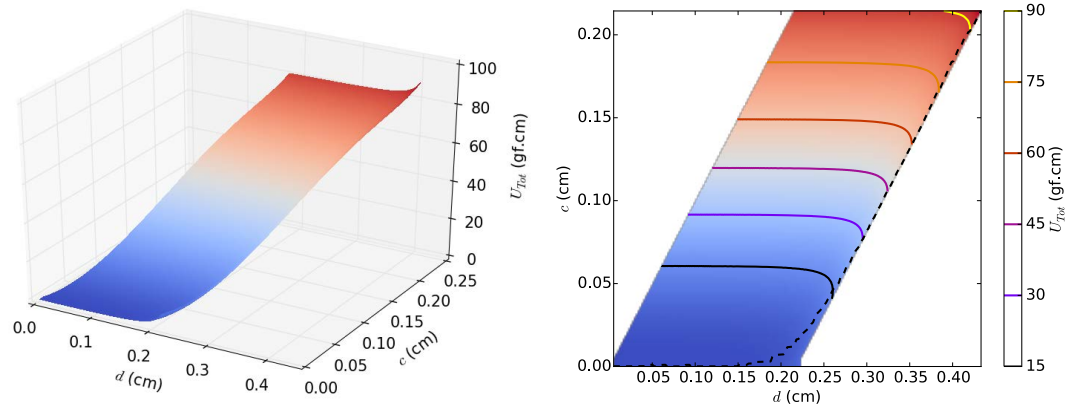
Figure 4.32: Minimised modified unit cell energy for a range of t_0.



(a) $t_0 = 0.05\,\mathrm{cm}$



(b) $t_0 = 0.15\,\mathrm{cm}$



(c) $t_0 = 0.25\,\mathrm{cm}$

Figure 4.33: Modified unit cell energy surfaces for a range of t_0, minimised along r'.

4.3.3 Discussion

In this section we have presented, examined, and compared two alternative models of a knoppy web unit cell. The two models both leverage the sphere model, but differ in their assumptions about the web component of the unit cell.

It is reasonably obvious from section 4.3.1.3 that the basic unit cell model is unrealistic. The geometric "jumps" do not ocur in real behaviour of knoppy web, and as pointed out in section 4.3.2, assumption 4.3.3 is fundamentally inconsistent with the underlying physics. The modified unit cell model offers significant advantages in both cases—it generates realistic compression behaviour and is a step towards more accurately simulating the non-uniform strain that occurs within the web component. However, it is still only an approximation of the knoppy web.

The true behaviour lies somewhere in the middle of these two approaches. In the basic model, the entire web component has uniform strain (fig. 4.34a); in the modified model, the web regions above and below the sphere take on a different strain to the web region around the sphere (fig. 4.34b). In reality, the strain in the web will be a continuum. When compressed around our simple knop model, web fibre above and below the inner region of the sphere would experience high strain, with the highest strain experienced near the boundary between the inner and outer sphere regions (because more web fibre is being compressed to the same height); the strain would then taper off towards the edges of the unit cell (fig. 4.34c). Compared to this, the basic unit cell model underestimates the developed strain, while the modified unit cell model overestimates it above and below the sphere.

To accommodate these strain gradients in a knoppy web model, we would need a model of fibrous assemblies that can describe anisotropic strain. None of the models described in section 3.3 can do this; thus it would be necessary to take an existing model and extend it to anisotropic strain. In some respects, it may actually be simpler to develop a new fibrous model specifically around the knoppy web geometry. This could then account for the effect of web fibre properties on the micromecanical structure; for example, the web fibre length being longer than the knop diameter means that fibres proximal to knops would tend to be oriented parallel to the knop surface. Such a model could be combined with our simple knop model to give a better approximation of the compression physics.

For the purposes of this thesis, however, we believe the modified unit cell model is sufficient to describe the behaviour of knoppy web to first order.

4.3.4 Simulating knoppy web

Extending the unit cell to a full knoppy web is simply a matter of scaling. By assumptions 4.3.1 and 4.3.4 we can imagine dividing the knoppy web into an i by j grid of columns, and each column into a series of k unit cells. In keeping with existing assumptions of affine deformation, we would define the overall compression of the knoppy web as

$$d_{\text{tot}} = kd, \tag{4.64}$$

and by the energy method, the overall strain energy of the knoppy web would be

$$U_{KW} = ijkU_T. \tag{4.65}$$

Thus, the knoppy web strain energy is directly proportional to the unit cell strain energy. We can therefore normalise the unit cell for comparison with actual knoppy web. The normalised compression distance $\tilde{d}$ is

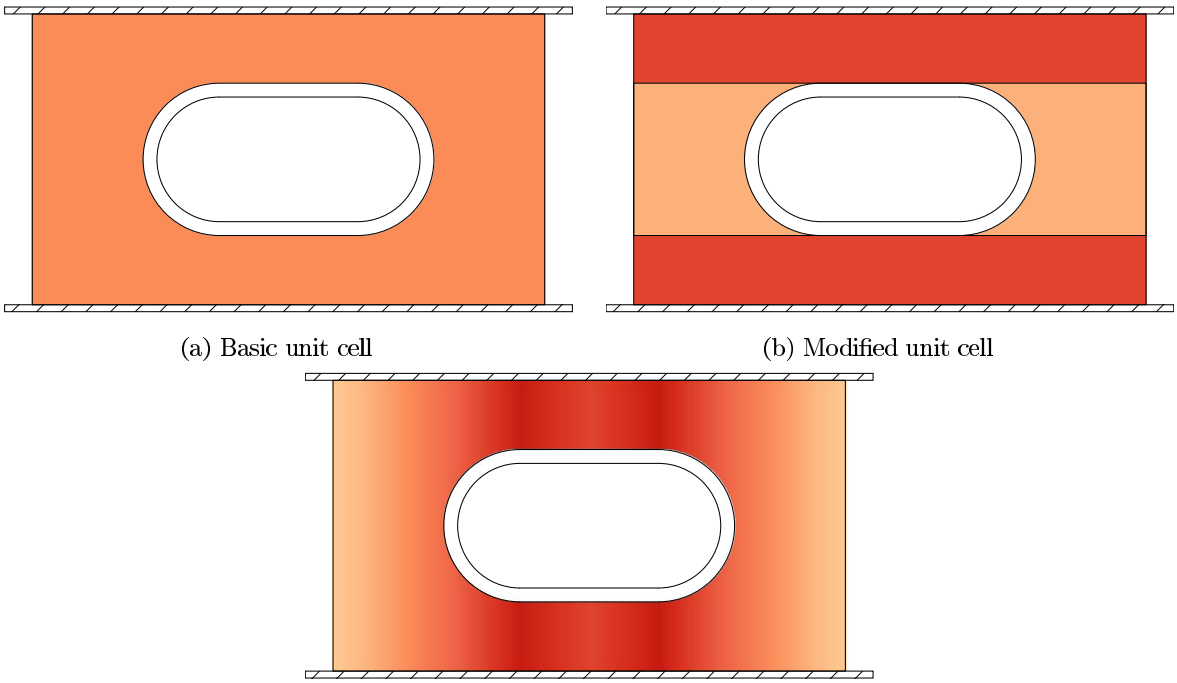
$$\tilde{d} = \frac{d}{r_0 + h + t_0}, \tag{4.66}$$

and the corresponding energy density $\tilde{U_T}$ is

$$\tilde{U_T} = \frac{U_T}{8(r_0 + h + t_0)^3}. \tag{4.67}$$

4.4 Summary

In this chapter we have developed the first theoretical model of a knop, approximating it with a spherical membrane. We have used this to develop a model describing to first order a unit cell of knoppy web.



(a) Basic unit cell

(b) Modified unit cell

(c) A more realistic gradient

Figure 4.34: Strain gradients within the web component during compression

Chapter 5

Experimental validation of knoppy web

5.1 Introduction

The overriding goal of this thesis has been to aid the product development process for knoppy web. In chapters 2 to 4, the focus was on the theory behind knoppy web and its production. In this chapter, we pivot to the experimental side, and present the results of the product development process.

Sections 5.2 and 5.3 show experiments conducted on knops to investigate their short-term and long-term resilience against compression. Section 5.4 describes an in-depth study of the effect of several production parameters on the properties of knoppy web. In section 5.5 we take the various knoppy webs obtained during product development, and simulate them with the knoppy web model developed in chapter 4. Finally, section 5.6 describes the development of optimum specifications for several knoppy web blends, and presents initial products and consumer trials.

Key original contributions in this chapter:

- A demonstration of the compression resilience benefits of knops over down (section 5.2.2).
- Optimum specifications for several knoppy web blends (section 5.6.2).

5.2 Resilience after compression of various loose fills

One of the key design parameters for knops is that they have greater resilience after compression than down. An experiment was conducted to quantify the difference in resilience between several types of loose fill:

- Down clusters.
- Goose feather.
- Micro-knops (22% high bulk wool, 61% fine wool, 12% micro fibre, 5% PLA).
- Untreated pure wool knops (27 µm wool).
- Underbody knops (50% SK merino, 35% cut WA12 blend, 15% 32 mm PLA) (these are the same knops used in section 5.3).

5.2.1 Method

The test for resilience after compression is a modified version of the standard bulk test for loose knops [87][1]:

[1] The pre-compression weight and the weight of the light disk are the actual weights used in the experiment, and deviate very slightly from TM 272 [87] due to wear and tear on the apparatus. Subsequent testers should use the correct weights.

1. 56.7 g of loose fill is evenly spread into a 19 cm diameter cylinder.
2. The fill is compressed under a weight of 170.8 g for 30 s.
3. The weight is removed for 30 s.
4. The above two steps are repeated once.
5. A 57.2 g disk is placed on the fill.
6. After 30 s, the height of the fill is measured.
7. The disk is left on for a further minute, during which photos are taken.
8. The disk is removed for 30 s.
9. The fill is compressed under a weight of 2231.0 g for 2 minutes.
10. The weight is removed for 30 s.
11. The 57.2 g disk is placed on the fill.
12. After 30 s, the height of the fill is measured.
13. Post-compression photos are taken.

One of the issues raised with this methodology was its applicability to down clusters and goose feather. To summarise the existing methodologies:

- The standard bulk test for knops [87] applies a 60 Pa pressure to a 56.7 g sample twice for 30 s, and then a 20 Pa pressure for 30 s.
- The standard bulk test for feather and down [88] applies a 14.8 Pa pressure to a 20 g sample for 60 s.
- This methodology follows the standard bulk test for knops [87], but subsequently applies a 770 Pa pressure for 120 s, and then a 20 Pa pressure for 30 s.

The key difference is the use of pre-compression when measuring the bulk, which is retained in this methodology. Feathers and down are never measured with pre-compression, and therefore record higher initial bulks. It was argued (by a potential client) that this methodology does not provide a "fair" comparison. We disagree with this assertion, for two reasons:

- Down never exists in products in an uncompressed state. As a loose fill that must be trapped in place within products (usually via quilting), there is always a minimum pressure applied to the down.
- This methodology is intended to give a simple indication of the relative properties of loose fills in compression-oriented products, such as pillows or sleeping bags (the latter of which must act both as an overbody and underbody product). Under realistic usage conditions, these products are rarely returned to their minimum-pressure state before each use. The exception to this may be pillows, which consumers are aware need to be regularly fluffed if they contain loose fill—but the results obtained by this methodology should correlate with the necessary frequency of fluffing.

We assert that the inclusion of a pre-compression step therefore gives results that are more reflective of user experiences. Furthermore, if the pre-compression step were to be left out, then the down clusters would certainly have a higher initial bulk than the knops, but would also have a higher bulk loss.

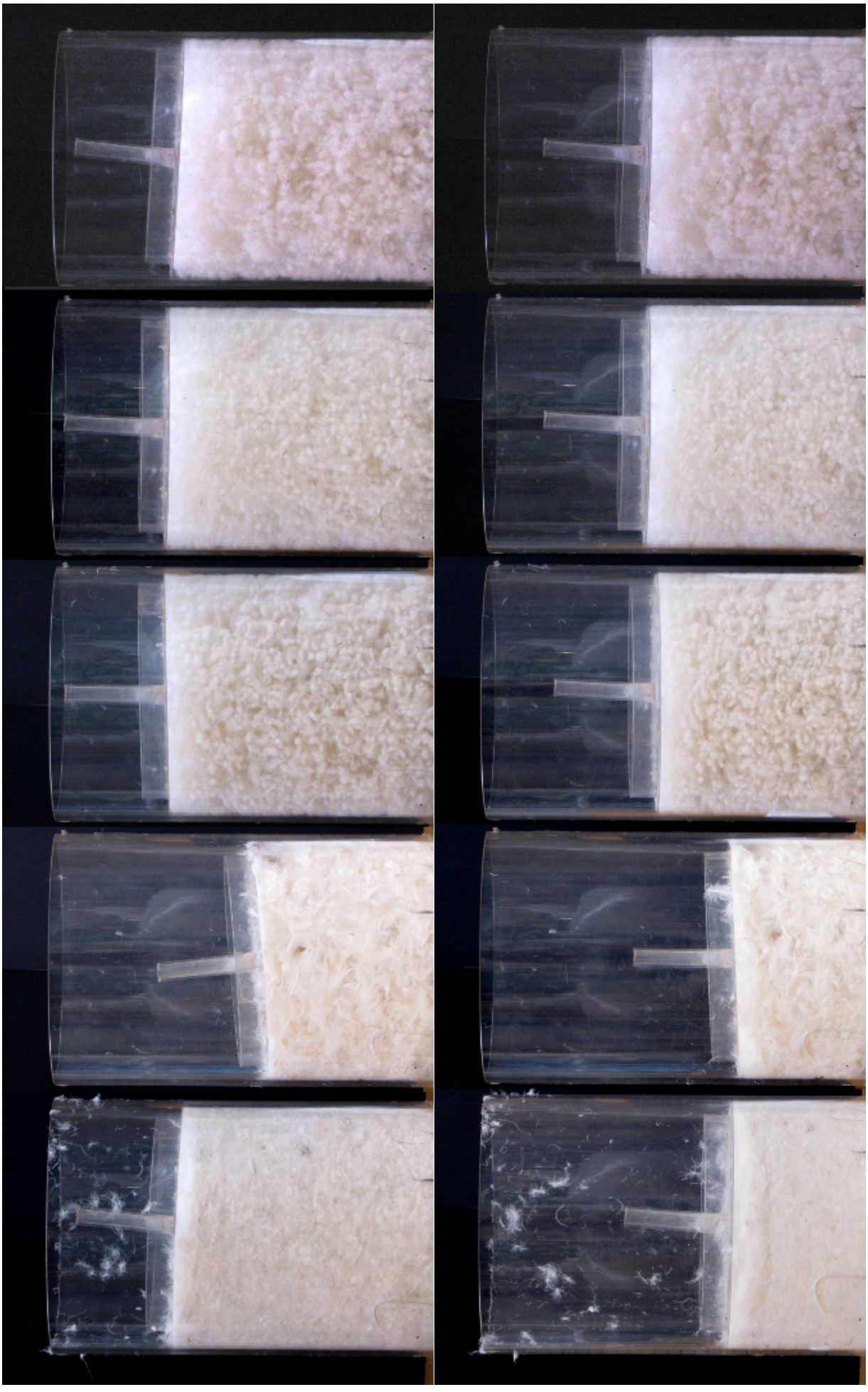


Figure 5.1: The various fills pre- (top) and post-compression (bottom). From left: down, goose feather, micro-knops, 27 µm wool knops, underbody knops.

Table 5.1: Pre- and post-compression bulks. The micro-knop bulks are calculated with a sample weight of 57.4 g instead of 56.7 g.

Fill type	Pre-compression bulk ($cm^3\,g^{-1}$)	Post-compression bulk ($cm^3\,g^{-1}$)	Bulk loss
Down	100.9	58.4	42.1%
Goose feather	69.5	51.5	25.9%
Micro-knops	105.5	83.0	21.3%
27 µm wool knops	107.6	88.8	17.5%
Underbody knops	104.0	89.8	13.7%

5.2.2 Results

Figure 5.1 shows the various fills after the initial bulk test, and after applying compression. Table 5.1 shows the corresponding measured bulks. The measured bulks for the micro-knops have been adjusted, as they had been bonded an hour previously and were still recovering weight. There was 57.4 g in the cylinder after testing, up from the initially-weighed 56.7 g.

It is clearly evident from fig. 5.1 that the knops withstand compression considerably better than the down clusters, with 2–3 times less bulk loss. The knops also perform better than the goose feather, even though their inital bulk was considerably higher. The knops all have approximately the same initial bulk as down for the same weight, indicating that they will exhibit similar loft in end products.

Within the knops, several points are evident:

- The underbody knops have the least bulk loss. This is expected—they are larger, use a bulkier wool blend, and have a higher PLA content.
- The micro-knops have the most bulk loss (although only half as much as down), due to a low fraction of high-bulk wool and a lower PLA content. This blend also results in a softer feel than the other knops; for these two reasons, these knops would be best suited to apparel products.
- The 27 µm pure wool knops perform better than the micro-knops, which we attribute to the fact that the fibres in the 27 µm knops are coarser, and store more compressional energy. However, the 27 µm knops have no PLA fibres to inhibit felting, and so under repeated compression they would be expected to lose their advantage.

5.3 Long-term compression of knops

One of the tangible benefits of down is that it can be compressed to high levels for extended periods of time, and once uncompressed its bulk can be nearly completely recovered simply through additional processing [89]. The agitation caused by e.g. blowing the down into products is sufficient to separate the down clusters [20]. If knops can withstand similar treatment, then it becomes feasible to produce knops in volume, bale them, and store them until they are required for knoppy web production.

This would also improve the scalability of knoppy web production within New Zealand. The cost of shipping knoppy web internationally is not economic without compressing the knoppy web, to increase the weight of product that can be fitted into a shipping container. Knoppy web is produced as rolls, which are difficult to compress. The only feasible option is to vacuum-pack the rolls, but this only increases the density by 2–3 times. At this density, shipping costs from New Zealand to the US are approximately $3 per kg. This is manageable in the short to medium term, but will ultimately encourage production closer to the point of consumption. If knops can withstand high compression for long periods, then the shipping cost per kg could be reduced significantly. The knoppy web blend could then be produced locally, and shipped overseas for airlaying and bonding.

Table 5.2: Measured bulks of the knops prior to compression

Knop type	Bulk
Unbonded	$104.0\,\mathrm{cm}^3\,\mathrm{g}^{-1}$
Bonded	$105.1\,\mathrm{cm}^3\,\mathrm{g}^{-1}$

One of the important factors is how long it takes for the knops to recover their bulk. Dry wool at room temperature shows stress-relaxation and creep [8], but this can be reversed by immersion in water [7], or raising the temperature above the glass transition temperature [9]. We propose that dry steam can be used to quickly recover the bulk of the knops.

Another important factor is whether the knops should be bonded before compression, or can be left unbonded. Bonding of knops on their own is not something that has been done at production-level quantities before, and is an additional processing step that would require considerable design effort and cost outlay. If unbonded knops can withstand high compression as well as bonded knops can, then the existing production line can be used as-is, with bonding carried out at the export destination once the knops are blended and airlayed as knoppy web.

In this section, we present an experiment conducted to examine the resilience of both bonded and unbonded knops under long-term compression. We examine the evolution of bulk post-compression both as-is and with steaming. We show that steaming is an effective method to recover the bulk of compressed knops, and that there is no obvious benefit to bonding the knops before compression.

5.3.1 Method

The knops for this experiment had the following component ratios by weight:

- 50% SK merino.
- 35% WA12 blend cut.
- 15% 32 mm PLA (4 Denier).

1.8 kg of knops were provided for the experiment. 736 g was split into five groups of about 150 g; each of these was then bonded in a Moffat bonding oven at 130 °C for 5 min. Each group was teased apart by hand immediately after bonding, to minimise the degree of inter-knop bonding.

Prior to compression, bulk measurements were taken for the bonded and unbonded knops using the standard bulk measurement process for knops [87]. Table 5.2 shows the measured bulks. The bonded knops are slightly more bulky than the unbonded knops, which is expected—the bonded PLA substructure aids the knops' recovery after the light pre-compression performed during the bulk test. However, the near-negligible difference means that unbonded and bonded knops can be compressed by the same assembly and achieve the same compression ratio.

5.3.1.1 Preparation

To ensure that an adequate comparison could be made, it was important that the bonded and unbonded knops were compressed evenly. To this end, a compression apparatus was designed whereby a single compressive force could be distributed nearly evenly across the knops. Six identical cans of diameter 152 mm were arranged in a hexagon, and each can had a "piston" consisting of an MDF circle and a length of cardboard tube (fig. 5.2). The cans were to be filled alternatively around the hexagon with bonded and unbonded knops; this ensured that any non-uniform distribution of the compression force could be later averaged out. A scale was attached to the side of each piston (fig. 5.3), so that during the experiment the height of the knops in each can could be measured.

Figure 5.2: The hexagonal assembly of cans, and the components for the pistons.

Figure 5.3: The fully-assembled can-and-piston compression aparatus.

Figure 5.4: Loosely filling a can with knops.

Figure 5.5: Can 5 being incrementally filled.

Table 5.3: Test compression of can 1.

Top weight (kg)	Height (cm)	Bulk ($cm^3\,g^{-1}$)
5.0	14.1	12.8
7.0	14.0	12.7
9.7	13.4	12.2
11.4	12.7	11.5
22.3	10.0	9.1

Each can was filled with 200 g of knops. Cans 1, 3 and 5 were filled with unbonded knops, while cans 2, 4 and 6 were filled with bonded knops. The knops were distributed evenly across the cross-section of the can in clusters of < 1 g (in accordance with the filling procedure specified in TM 272 [87]). A tube of brown paper was used to aid the filling process (fig. 5.4). The tube could hold about 100 g of loose-filled knops; from this point, the knops were incrementally filled by compressing with a piston plate down to 10 cm to 20 cm below the top of the cylinder, and then loose-filling up to the top, until all 200 g were in the cylinder (fig. 5.5).

Finally, the piston plate was compressed down below the rim of the can, and secured in place with pins (fig. 5.6). This pre-compression will have introduced differences between the cans, because all cans were filled and pre-compressed within 2.5 hours; however this effect is considered negligible given that the timescale of the experiment is much longer than the time difference, and the pre-compression force was lower than the actual compression force used in the experiment (discussed in the next section).

5.3.1.2 Compression

As the goal was to simulate the effects of baling the knops, the required compression ratio was calculated using standard baling parameters previously used for knops. An average bale has dimensions of 1.1 m x 0.7 m x 0.7 m, and there are on average 50 kg of knops per bale. This gives a bulk of $10.78\,cm^3\,g^{-1}$. For the knop bulks specified in table 5.2, this is a compression of 89.7%.

To establish the required compression weight, can 1 was compressed by placing a series of different top weights onto its piston. Table 5.3 shows the resulting heights and bulks. To achieve a bulk of $9.1\,cm^3\,g^{-1}$ for all six cans, a total compression top weight of 133.8 kg would be required, alongside the weight of the pistons. The weights available gave a total of 134.6 kg, so all were utilised in the experiment.

To begin the experiment, the cans were arranged into a hexagon, and a square board was placed on top of the pistons. The weights were then added one-by-one on top of the board (fig. 5.7). The board and pistons together weighed 6.5 kg, giving a total load weight of 141.1 kg. This equates to an applied pressure of

$$\frac{141.1\,kg \times 9.8\,m\,s^{-2}}{6\pi\left(\frac{15.2\,cm}{2}\right)^2} = 1.27\,N\,cm^{-2}.$$

The compression apparatus was left for 128 days. The heights of the pistons were recorded at several points during that time, including the start and end.

5.3.1.3 Recovery

At the end of the compression period (fig. 5.8), the weights were removed. The piston heights were recorded several times while the weights were being removed. The knops were emptied into bags, and then taken to AgResearch for subsequent testing.

A bulk measurement was made for the knops in each can, about 2 hours after the end of the compression period. Then about a third of the knops (half of the knops from cans 3 to 6) were open-steamed on a Hoffman press. The knops from cans 3 and 4 were steamed for 30 s (fig. 5.9), and the knops from cans 5 and 6 were steamed for 15 s. The steamed knops were left for an hour, and then their post-steaming bulk was measured.

Figure 5.6: Can 1 filled with 200 g of knops.

Figure 5.7: The compression assembly just after the weights were applied.

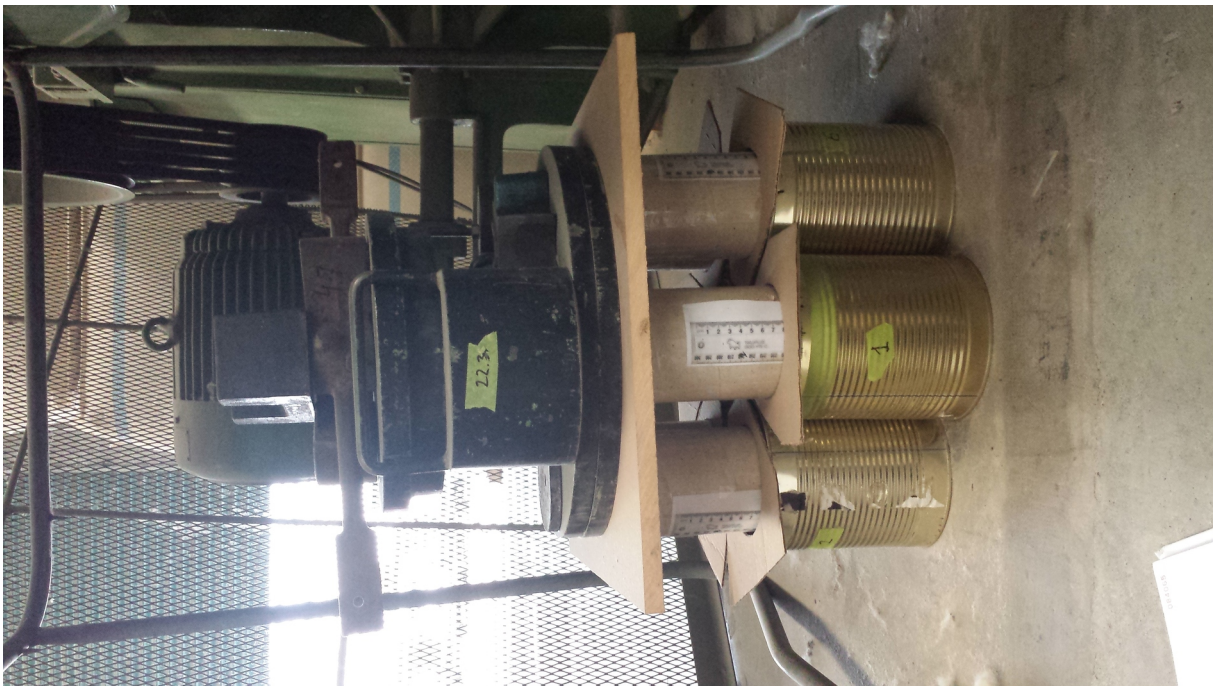


Figure 5.8: The compression assembly just before the weights were removed.

Figure 5.9: Knops from cans 3 and 4 being steamed for 30 seconds.

Unfortunately, at the time of decompression, the standard bulkometer for knops [87] was not available. Instead, a WRONZ loose wool bulkometer [90] was used to measure the bulk of the knops. It has several deficiencies compared to the standard knop bulkometer:

- The cylinder has a diameter of 8 cm, and is only designed to take a 10 g sample. This makes it difficult to distribute the knops evenly, compared to the standard bulkometer which has a diameter of 18.9 cm and uses a 56.7 g sample.
- The tester uses a 1.5 kg load for pre-compression and a 500 g load for bulk measurement. These are a factor of ten larger than the loads used in TM 272 [87], meaning that the post-compression bulks cannot be directly compared to the pre-compression bulks.

We could attempt to use the force-displacement curves for the knops post-compression to generate a mapping, but it would not be reliable because the bulk test includes pre-compression that the force-displacement curves do not. Instead, we re-measured the bulk of leftover uncompressed knops with the WRONZ loose wool bulkometer, and used this as the comparison point.

5.3.1.4 Force-displacement curves

The force-displacement curves were measured two weeks after the removal of compression and steaming. All knops were left under standard conditions (20 °C, 65% relative humidity) during this time.

To measure the force-displacement curves for the knops, the WRONZ loose wool bulkometer (with an internal cross-sectional area of 50 cm^2) was attached to an Instron 4204 [91] fitted with a 10 kg load cell (fig. 5.10). The following methodology was used:

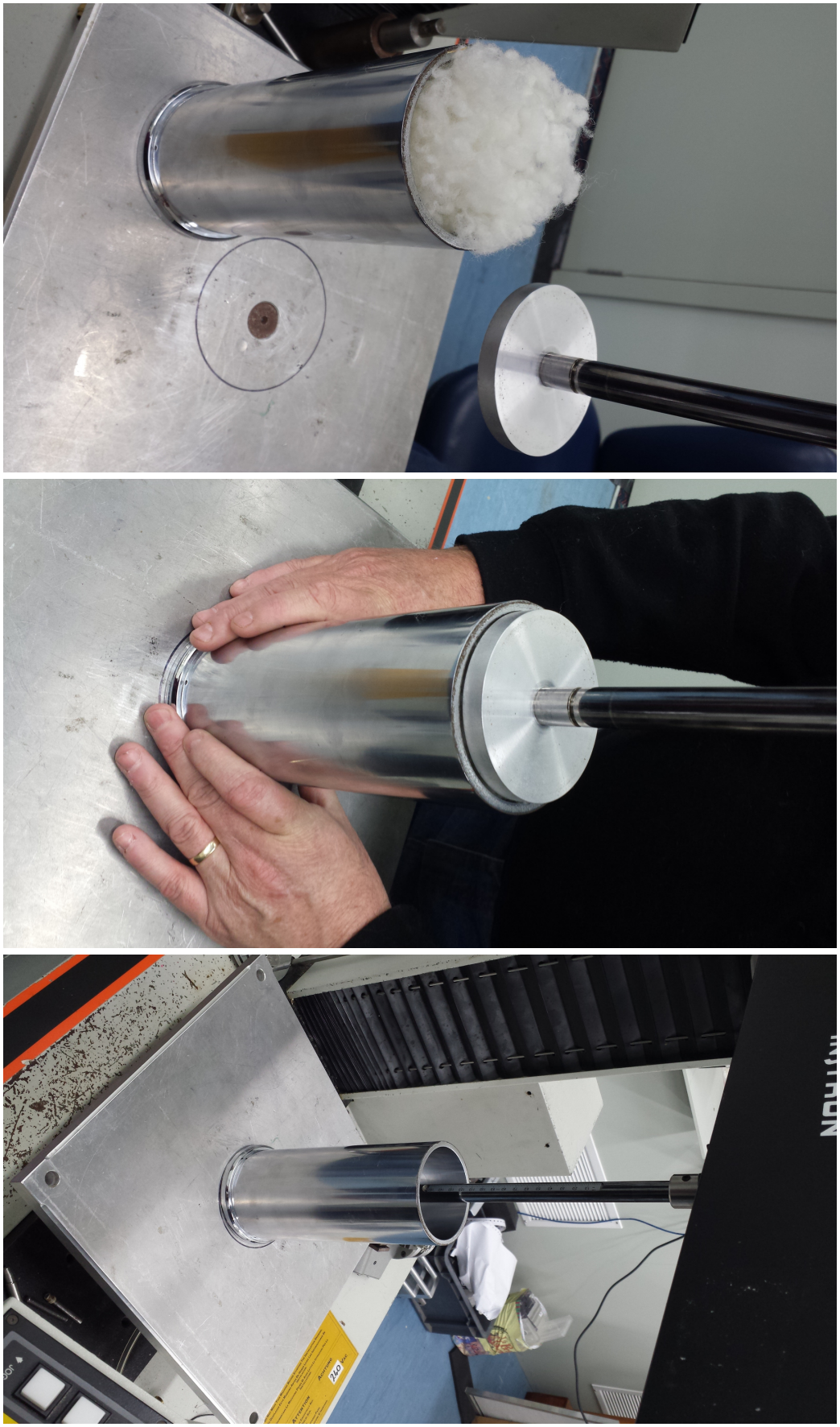
1. The compression chamber was filled loosely with 10 g of knops (fig. 5.10a).
2. The knops were lightly pre-compressed to the top of the chamber (fig. 5.10b). This was necessary to prevent knops from catching between the compression piston and chamber wall, and to provide uniform lateral containment for the duration of compression. This step was only necessary for knop samples that overflowed the compression chamber (which was most of them). The initial loading caused by this step was negligible (less than 10 g).
3. The compression piston was lowered into the chamber at a rate of 50 $mm\,min^{-1}$ (fig. 5.10c) until a load of 2.5 kg was reached, and then raised at the same rate until zero load was registered.

Despite the use of the non-standard compression chamber, the observed behaviour will be much more accurate than the measured bulk values, because the force is a controlled value in this phase, and the only discrepancy that is not a numerical factor is the distribution of knops in the compression chamber.

5.3.2 Results

5.3.2.1 Compression

Figures 5.7 and 5.8 show the compression assembly at the start and end of the experiment. It is evident that some slumping occurred; this can be seen more clearly in fig. 5.11, which shows the evolution of the piston heights over the duration of the experiment. The slumping was most likely due to the center of mass of the pile of weights not being perfectly centred on the board. A few of the smaller weights were moved around at the 30 day mark in an attempt to correct this (e.g. the 1.7 kg weight moves from the left side of fig. 5.7 to the back-right of fig. 5.8), and the effect can be seen in fig. 5.11: the board tilted away from cans 3 and 4, towards cans 1 and 6. It was not possible to reposition the larger weights (to correct the primary tilt from can 5 to can 2) without disrupting the experiment. However, the tilt itself is essentially negligible given the already significant compression levels. It can be averaged out due to the



(a) The filled compression chamber (b) Pre-compressing the knops to fit under the piston (c) Measuring the compression curve

Figure 5.10: Apparatus and procedure for measuring the force-displacement curves of knops.

Table 5.4: Measured pre- and post-steam bulks of the unbonded (1, 3, 5) and bonded (2, 4, 6) knops. The percentage values for pre- and post-steam bulks are percentages of the measured bulks of leftover knops that had not been subjected to long-term compression (unbonded: $34\,cm^3\,g^{-1}$, bonded: $38\,cm^3\,g^{-1}$). The steam regain percentages are increases over the corresponding pre-steam bulks.

Can	Steaming period (s)	Pre-steaming bulk ($cm^3\,g^{-1}$)	(%)	Post-steaming bulk ($cm^3\,g^{-1}$)	(%)	Steam regain (%)
1	—	23.5	69.1	—	—	—
2	—	21.0	55.3	—	—	—
3	30	22.5	66.2	32.5	95.6	44.4
4	30	20.5	53.9	33.0	86.8	61.0
5	15	22.5	66.2	31.5	92.6	40.0
6	15	20.0	52.6	32.8	86.3	63.8

alternation of bonded and unbonded knops (described in section 5.3.1.1). On average, the knops were compressed to 7.4% of their original bulk; the tilt caused a variance of ±0.9%.

5.3.2.2 Recovery

Figure 5.12 shows the heights of the pistons as the weights were removed after the compression period. Despite having been held at unequal heights for most of the compression period, the cans of unbonded knops sprang back to nearly identical heights, as did the cans of bonded knops. This bolsters the earlier assertion that the effect of the uneven pressure was negligible. Far more interesting, however, is the observation that the bonded knops sprang back significantly less than the unbonded knops. Even after subsequent removal of the pistons, with no remaining weight, the unbonded knops (fig. 5.13a) were recovering more than the bonded knops (fig. 5.13b).

Figures 5.14 and 5.15 show the unbonded and bonded knops just after they had been removed from their cans. The bonded knops were visibly flattened, with very "sharp" edges. The layers of knops formed under compression also stayed "stuck" together, coming out of the can in large groups. The unbonded knops by comparison looked softer, and had a more rounded, elliptical cross-section. They also formed less stable layers that came apart slightly easier.

Figure 5.16 shows this difference more strikingly. Both bags contain 200 g of otherwise-identical knops, but the unbonded knops have regained significantly more bulk than the bonded knops. The bonded knops also visibly retain the layers formed under compression, whereas the unbonded knops more readily dispersed.

5.3.2.3 Steaming

Table 5.4 shows the measured bulks of the knops, before and after steaming. As observed in the previous section, the unbonded knops have consistently higher post-compression, pre-steaming bulks than the bonded knops. However, upon steaming, the bonded knops have significantly better regain, and end up slightly bulkier than the unbonded knops. This makes sense, as the unbonded knops have already undergone some regain, and therefore have less inherent strain energy to release.

As a percentage of their uncompressed bulk, the unbonded knops appear to have performed better than the bonded knops. However, this is somewhat deceptive, for two reasons:

- The bulks were measured with the incorrect bulk tester. Recall from table 5.2 that the original bulks of the bonded and unbonded knops were very similar. Under this heavier test the bonded knops have a higher bulk than the unbonded knops, because of the added resilience created by the bonding process. Therefore, the measured bulks of the unbonded knops would be higher relative to the bonded knops under the proper test.

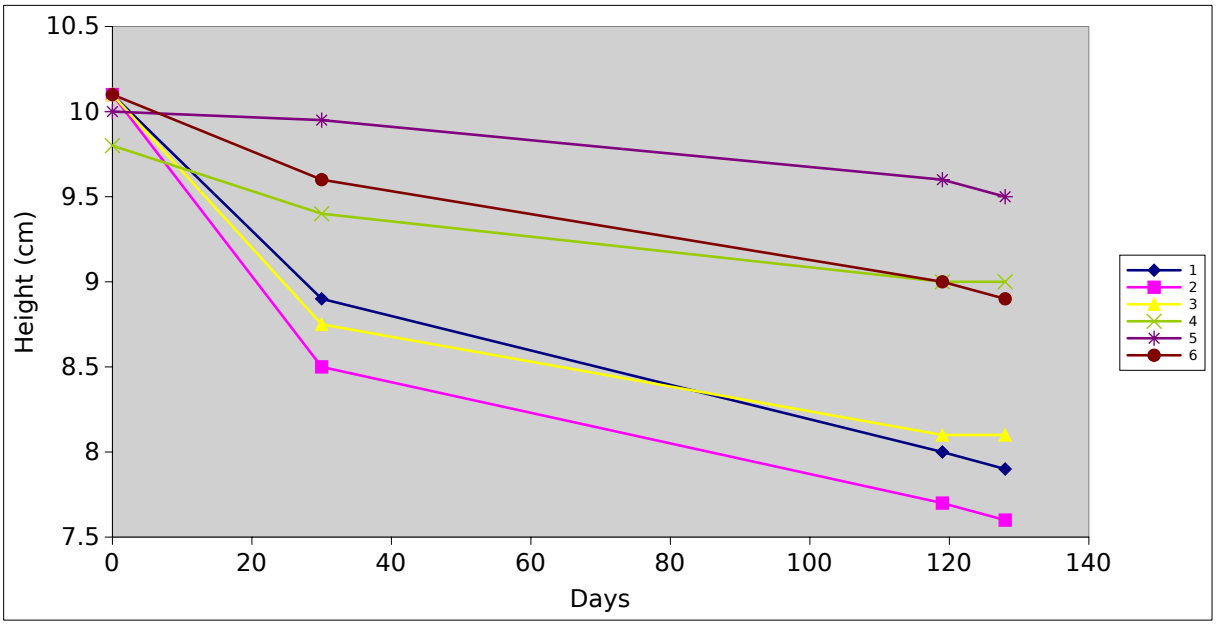


Figure 5.11: The evolution of piston heights over the duration of compression.

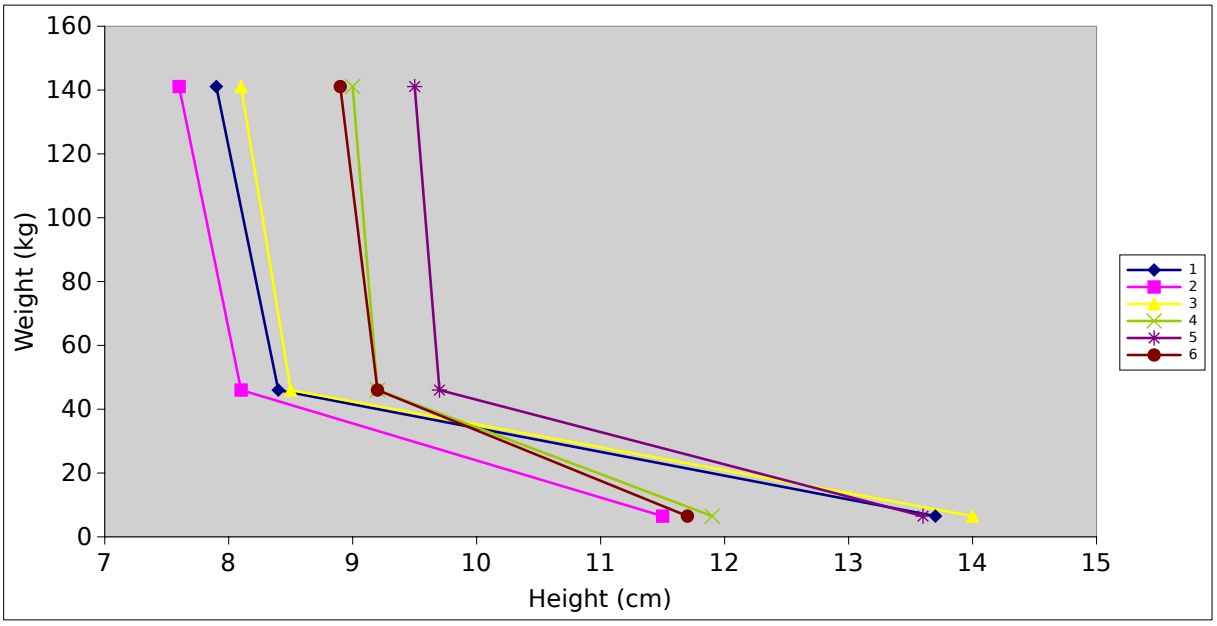


Figure 5.12: The evolution of piston heights as the compression weights were removed.

(a) Can 1—unbonded knops (b) Can 2—bonded knops

Figure 5.13: Knops after removal of the weight and pistons.

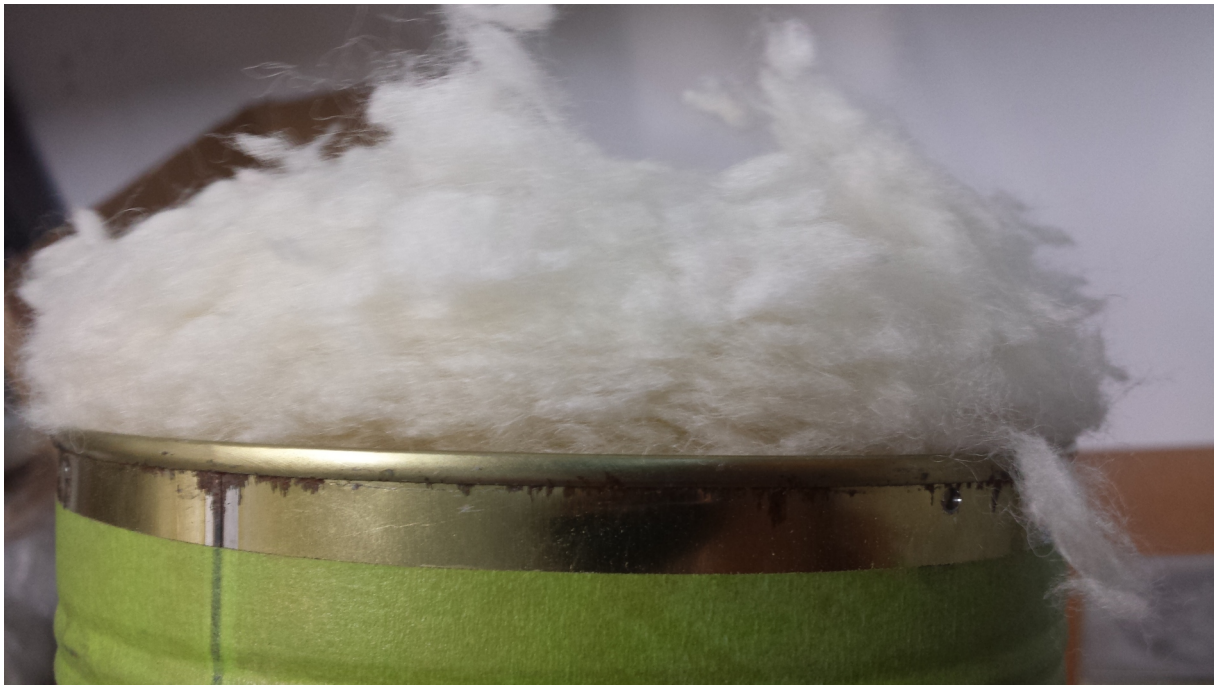


Figure 5.14: Side view of the unbonded knops from can 1 after removal of the weight and piston.

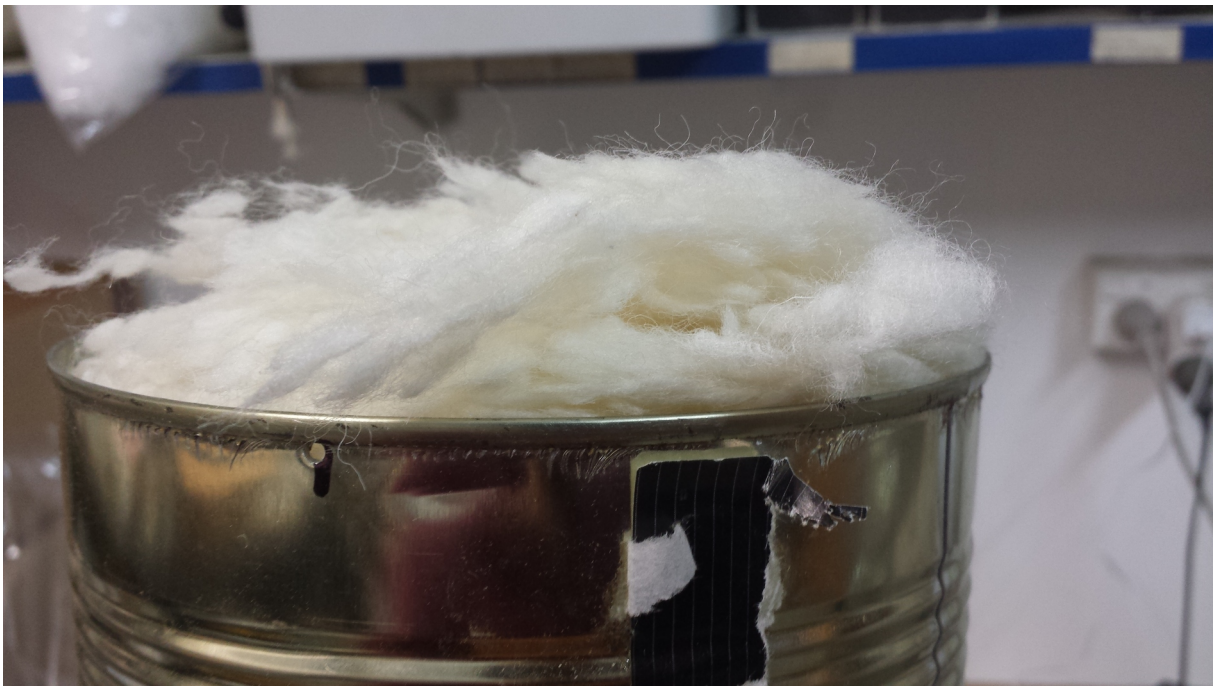


Figure 5.15: Side view of the bonded knops from can 2 after removal of the weight and piston.

Figure 5.16: Knops from cans 1 and 2 just over an hour after they had been emptied into bags.

- The knops were not conditioned prior to bulk measurement. The intention was to measure the bulks as soon after the end of the compression period as possible; the knops were therefore "conditioned" on the factory floor at slightly colder temperatures than standard[2]. The uncompressed knops had also been stored in the same factory building, and experienced similar conditions. However, the bulks of the steamed knops were measured only an hour after the steaming process, and effectively had been "conditioned" in a higher-temperature higher-humidity environment. Wool becomes less rigid as its water content increases [94], and the recent steaming could easily lessen the differences between the unbonded and bonded knops.

There was little benefit to steaming the knops for 30 s. The knops from cans 3 and 4 were observed to stop visibly expanding after approximately 15 s on the Hoffman press, and the similarity of the post-steaming bulks suggests that 15 s was the optimal time for steaming in this particular configuration. However, a scaled-up production line might require a longer steaming period than this (or not), depending on the specifics of the steamer and the thickness of knops through which the steam needs to permeate.

5.3.2.4 Force-displacement curves

Figure 5.17a shows the force-displacement curves for the leftover (pre-compression) knops. The effect of bonding is clear: the bonded and unbonded knops start and end about the same height (bulk), but the bonded knops can better withstand the applied strain.

After subjecting the knops to long-term compression, this behaviour is reversed (fig. 5.17b). The bonded and unbonded knops offer the same level of support at high strain, but the unbonded knops have a higher starting bulk than the bonded knops, and recover to a higher bulk.

Figures 5.18 and 5.19 show the force-displacement curves for the knops from cans 3 to 6. The steaming process is clearly a success—the steamed knop samples have nearly-identical compression curves to the uncompressed samples. There is some loss in recovery potential across all samples, which is to be expected given the magnitude and duration of the strain applied during the compression period.

Looking more closely at the force-displacement curves for the steamed knops, the unbonded knop samples exhibit slightly better performance than the uncompressed unbonded knops. The steam temperature was below the 130 °C necessary to melt the outer layer of the PLA (section 1.2); nevertheless, it is possible that some slight bonding occurred during the steaming process that resulted in a slight strengthening of the unbonded knops.

5.3.3 Discussion

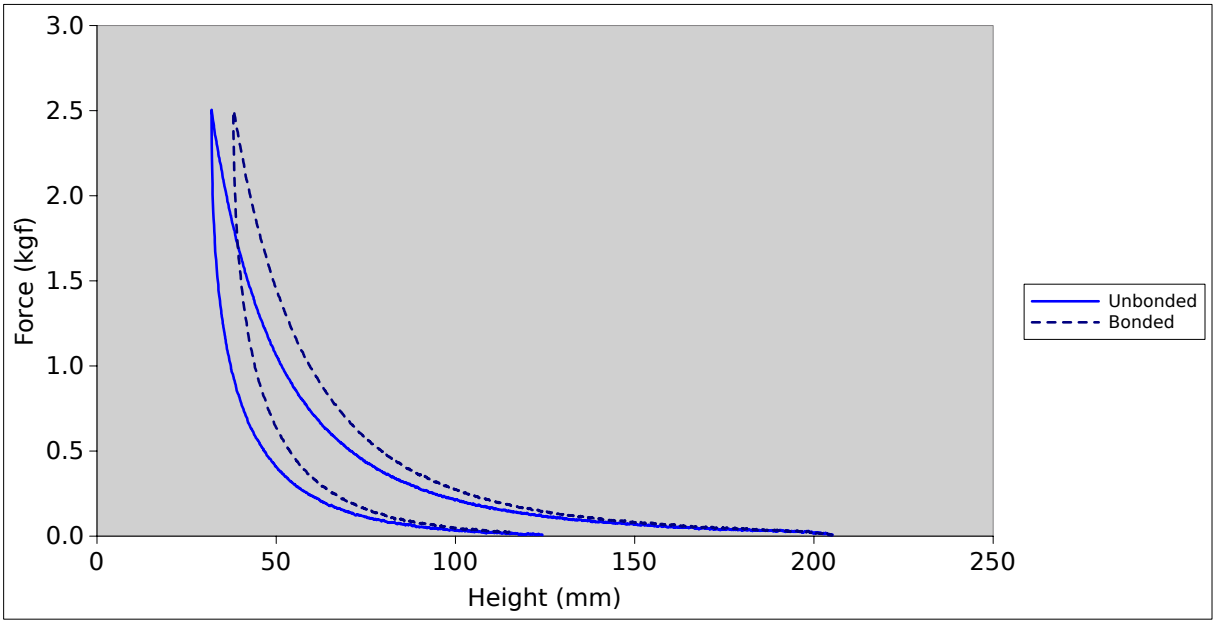
Two key observations were made in this experiment:

- Unaided, unbonded knops recover more than bonded knops.
- With steaming, both bonded and unbonded knops can recover nearly all of their previous bulk.

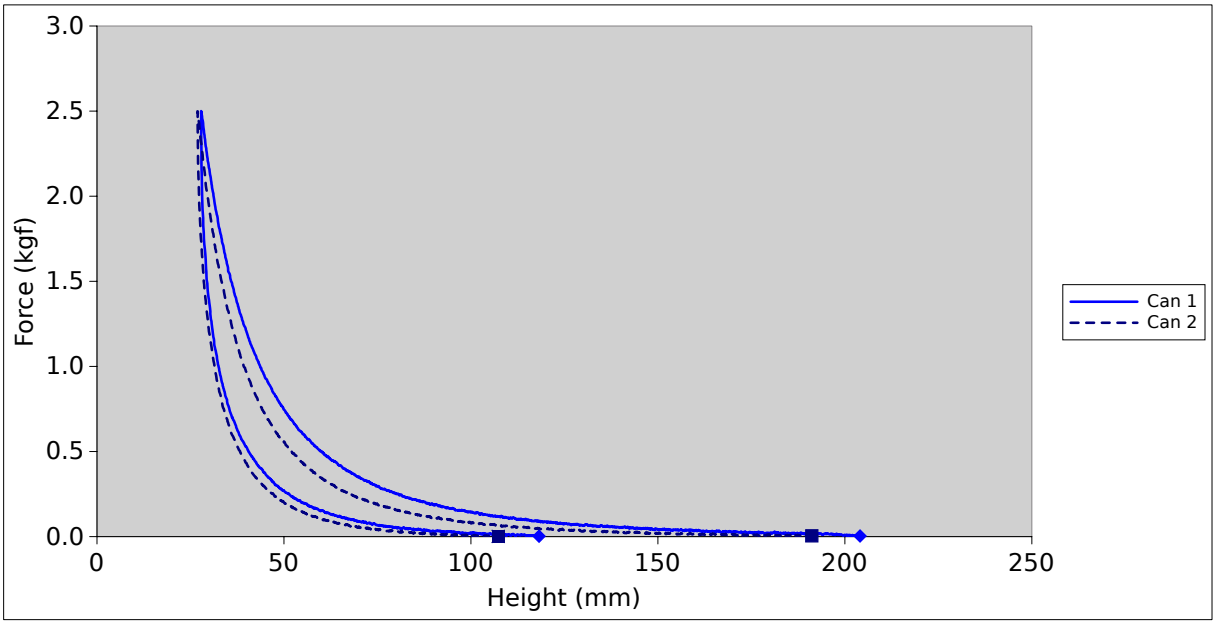
The fact that unbonded knops recover better after long-term compression than bonded knops is counter-intuitive. After 124 pages making the point (among others) that bonded knops have better resilience than unbonded knops, why should the opposite now be true? We propose that the reason for this difference is the PLA's effect on fibre movement within the knops.

As described in section 2.8, the wool fibres in bonded knops are spatially fixed by the presence of the bonded PLA substructure, meaning that during compression there is more bending of fibres and less slippage. When taken to extreme strains (such as in this experiment), the spherical knops are completely flattened, causing a relatively small fraction of the total fibre length (around the edges of the knops) to

[2]The compression period was from late autumn to early spring. On the day of de-compression, the maximum temperature was 19 °C [92] and the relative humidity was 65% [93].

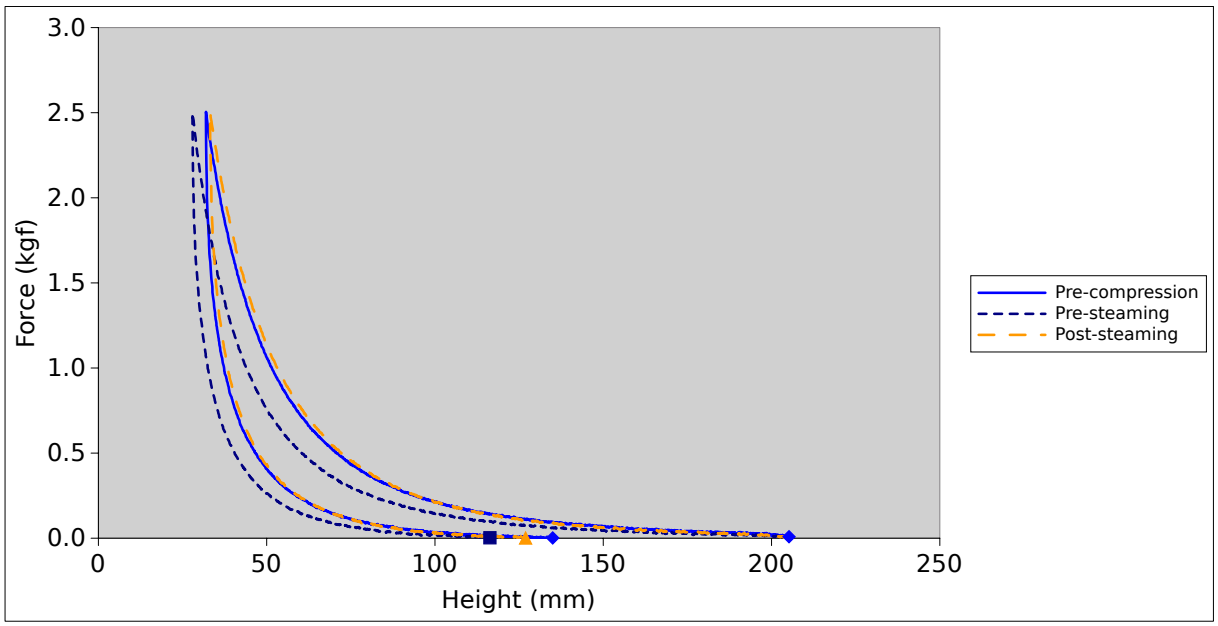


(a) Unbonded and bonded leftover knops

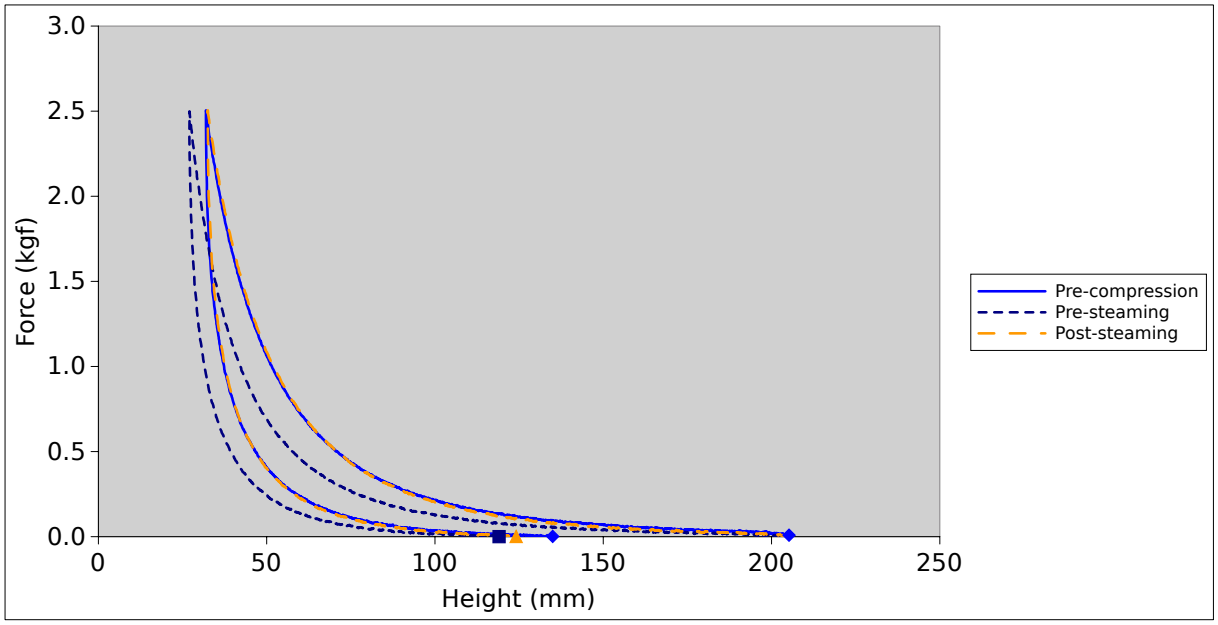


(b) Cans 1 and 2

Figure 5.17: Force-displacement curves for the unsteamed knops.

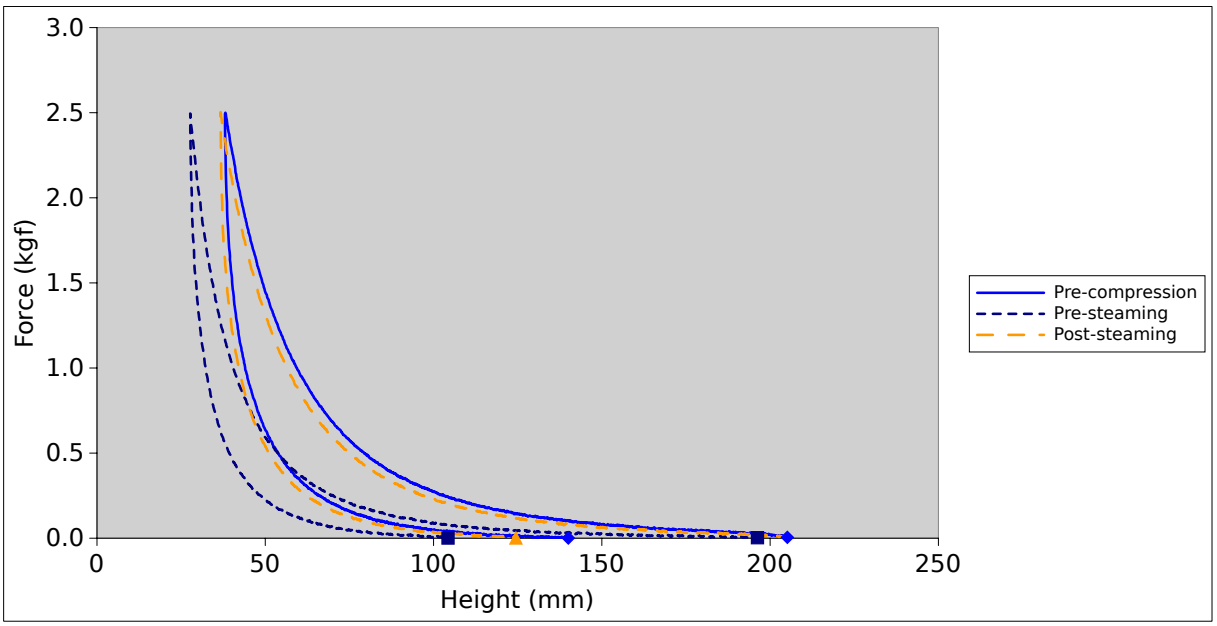


(a) Can 3 (30 s steam)

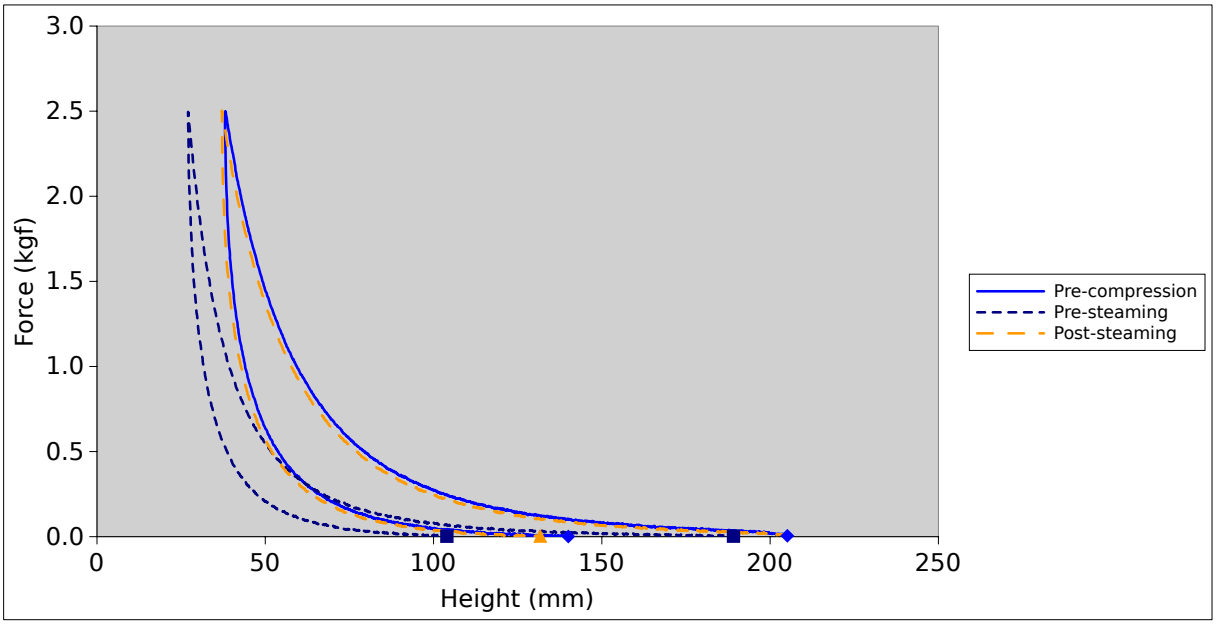


(b) Can 5 (15 s steam)

Figure 5.18: Force-displacement curves for the unbonded knops. The pre-compression data is from leftover uncompressed knops.



(a) Can 4 (30 s steam)



(b) Can 6 (15 s steam)

Figure 5.19: Force-displacement curves for the bonded knops. The pre-compression data is from leftover uncompressed knops.

take up extreme bending strain. Stress-relaxation [8] would readily occur in fibres being held in tight 180° bends for several months. This would reduce the stored energy and therefore the unaided recovery potential, resulting in "flatter" knops (fig. 5.15).

In the case of the unbonded knops, the wool and PLA fibres are not constrained by bonds and can move more freely. This could allow the fibres within the knop wall to reconfigure themselves as the knop is taken to extreme compression, enabling the bending strain to be spread over a larger fraction of the total fibre length. Thus the bending stress relaxation will be lower, and the knop will more readily recover its bulk after removal of the compression force.

Besides improving the resilience of the knops, the bonded PLA substructure helps to reduce the felting ability of the wool fibres in the knops (section 2.3.2.1). One concern with not bonding the knops before compression was that their properties would be compromised, due to felting. This concern is clearly unfounded based on the results shown in section 5.3.2.4. We suggest that the reason for this is that there is only one compression-recovery cycle between manufacturing the knops and steaming them. Furthermore, the extreme compression ratio creates considerable static friction between and within the knops, greatly inhibiting the ability of the fibres to respond to external agitation. Felting requires that the wool fibres progressively ratchet past each other; without repeated compression or agitation, this cannot happen.

In summary, knops can be compressed to high strains for several months, and be successfully recovered with steam. Furthermore, there is no benefit to bonding the knops before compression, and therefore no need for the added costs of adding knop bonding to the production line. The knoppy web can be made and bonded at the destination using knops and fibre shipped from New Zealand in dense bales.

5.4 Variations of the knoppy web blend

Having established the tangible benefits of knops, the next key step is to determine their optimal usage in knoppy web. As outlined in section 2.2, there are several production parameters that can be used to alter the knoppy web blend. An experiment was conducted in which several of the parameters were varied across a series of trial production runs. The objective of this experiment was to determine how varying the knoppy web blend affects its measurable properties.

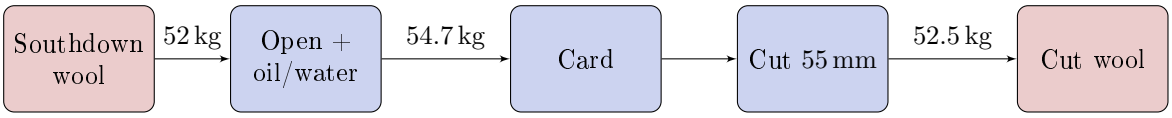
5.4.1 Overview of Lots

The number of variables that could be examined, and the number of variations that could be explored, were limited by the cost of factory runs. Three variables were chosen for variation in this experiment:

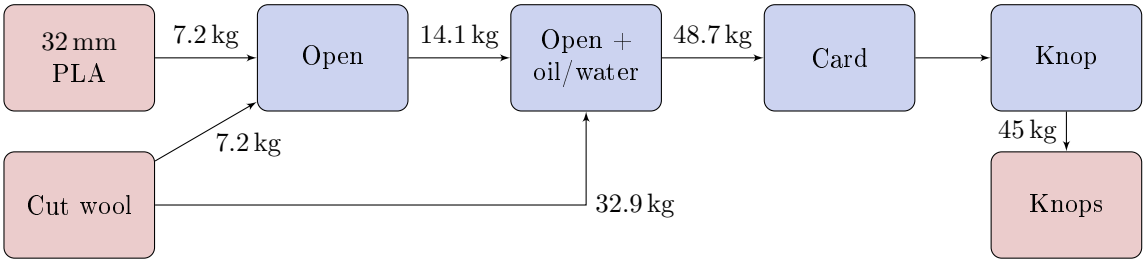
- The ratio of knops to web.
- The ratio of PLA to wool within the web.
- Whether the knoppy web blend was carded prior to airlay.

All other parameters of the knoppy web blend were held constant (as much as feasibly possible). In particular, a single type of knop was used for all Lots, comprising 85% 55 mm-cut Southdown wool ($29.7\,\mu m, 28\,cm^3\,g^{-1}$) and 15% 32 mm bicomponent PLA (4 Denier).

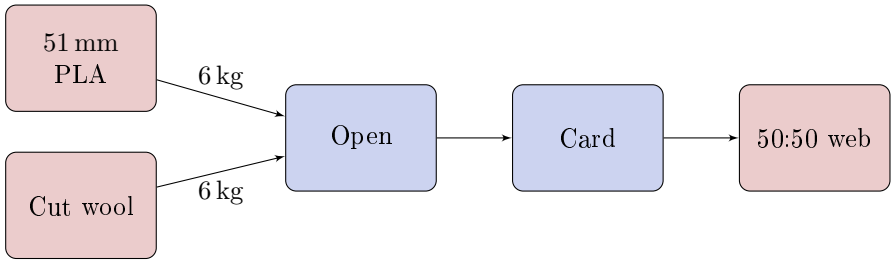
Table 5.5 outlines the six Lots considered in this experiment. Lot 2 was the "control" Lot, and formed the midpoint of the variations. Lots 1 and 3 were variations on the ratio of knops to web in the blend. Lots 4 and 5 altered the ratio of PLA to wool within the web. Finally, Lot 6 was the same blend as Lot 2 but underwent an additional carding step to further blend the knop and web components.



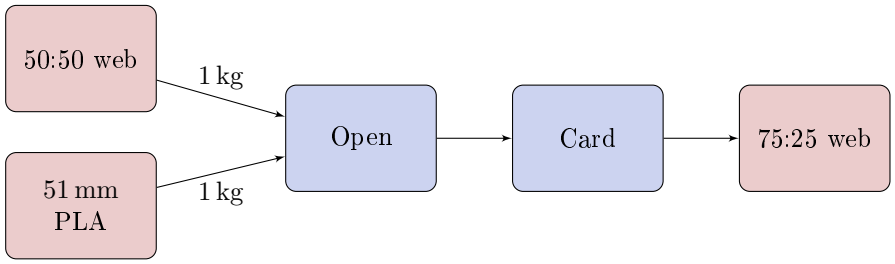
(a) Initial wool processing



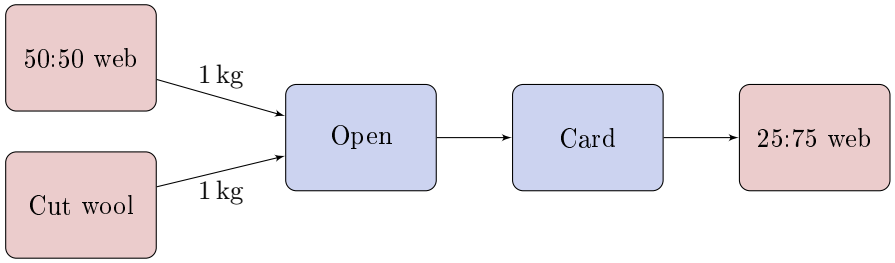
(b) Knop production



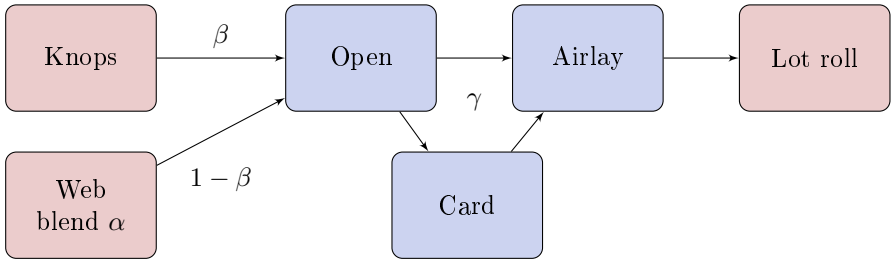
(c) 50:50 web preparation



(d) 75:25 web preparation



(e) 25:75 web preparation



(f) Production of a lot with web PLA:wool ratio α, knop:web ratio β and optional extra carding (γ).

Figure 5.20: Production process for the knoppy web lots.

Table 5.5: Overview of the various Lot variations, and the total wool fraction for each Lot.

Lot	Knop:Web	PLA:Wool (in web)	Carded?	Wool fraction
1	90:10	50:50	No	0.815
2	80:20	50:50	No	0.78
3	70:30	50:50	No	0.745
4	80:20	75:25	No	0.73
5	80:20	25:75	No	0.83
6	80:20	50:50	Yes	0.78

5.4.2 Methods

5.4.2.1 Manufacture of Lots

The Lots were manufactured in several stages. Steps were performed in bulk as much as possible to better simulate industrial blending processes. The blending was calculated on 12 kg Lots, reduced to 10 kg after processing losses; however Lot 6 was added after Lot 3 had been manufactured, and so the remaining Lots were made in 9 kg batches.

The first step was to prepare the bulk wool, via the process shown in fig. 5.20a. 52 kg of Southdown wool (29.7 µm, 28 $cm^3 g^{-1}$) was passed through the opener (with 1 gallon of oil/water added), resulting in 54.7 kg of opened fibre. This was carded, and then cut with a set blade length of approximately 55 mm to remove the longer fibres (as discussed in section 2.4), giving 52.5 kg of cut wool fibre—the base wool stock for the remainder of the experiment.

Knop preparation The knops for this experiment were 85% cut wool, and 15% 32mm bicomponent PLA. Due to the relatively small size of the batch, the knops were blended in two stages to give a more even distribution. Figure 5.20b shows the knop production process:

1. A 50:50 blend was created by passing 7.2 kg each of cut wool and 32 mm PLA through the opener.
2. The 50:50 blend was then blended 70:30 with cut wool by passing them through the opener (with oil and water), giving 48.7 kg of 85:15 blend.
3. The 85:15 blend was carded and then knopped.

45 kg of knops were produced, with an indicative measured bulk [87] of 79 $cm^3 g^{-1}$. This would appear to be a huge increase in bulk from the 28 $cm^3 g^{-1}$ of the plain wool, but the numbers cannot be directly compared—recall from section 5.3.1.3 that the bulk of plain wool is measured with a different methodology that employs much heavier weights [90].

Web preparation The web used in this experiment was a blend of the stock cut wool and 51 mm bicomponent PLA. Three blend ratios were used in this experiment—50:50, 75:25 (more PLA) and 25:75 (more wool).

The 50:50 web blend was created by passing 6 kg each of cut wool and PLA through the opener, followed by carding (fig. 5.20c). The other two blends were created by passing 1 kg of the 50:50 web blend and 1 kg of either PLA or cut wool through the opener and card (figs. 5.20d and 5.20e).

Knoppy web blend Figure 5.20f shows the process used to make the various Lots. The knoppy web blends were created by passing the appropriate ratio of knop to web (of the correct web blend) through the opener. This gave a reasonable distribution, though in the small (9 kg) loads being made in this experiment the blending was suboptimal.

Each knoppy web blend was then run through the DOA airlay to manufacture the final knoppy web. The output was rolled by hand onto cardboard tubes as it emerged, and the layers separated with a light non-woven fabric.

5.4.2.2 Bonding

The production line used for the experiment did not include a continuous-flow bonding oven, so the Lots could not be immediately bonded as they came off the airlay. Instead, samples required for testing were taken from the unbonded rolls and were bonded individually.

Sheets were cut from the rolls and then bonded in a Moffat oven. The oven could comfortably hold sheets of dimension 600 mm x 420 mm, and most sheets were cut to dimensions within this. However, it was noted that the sheets underwent (often considerable) shrinkage during the bonding process. In order to ensure that adequately-sized bonded sheets were obtained for creating test samples, some unbonded sheets were cut at 600 mm x 500 mm and curved at the edges to fit into the oven.

Indicative results for shrinkage by area were obtained by taking the dimensions of similar sheets and averaging them to obtain the average reduced dimensions for each Lot.

5.4.2.3 SEM

For general bedding products, the most important tests are at a macroscopic scale (see the following sections). However, the unique value of the knoppy web is in the presence of the knops and their interaction with the web. Examining the Lots at a microscopic scale can reveal important and interesting details about both the blending and bonding stages of production.

Representative samples of both knop and web were taken from the edges of the bonded sheets; these were mounted and coated with carbon. A Scanning Electron Microscope (SEM) was then used to examine the samples for the presence of bonds. Specifically, the JEOL Field Emission-SEM in the Mechanical Engineering depertment at the University of Canterbury was used. Wool fibres can be identified by their scaly structure, and PLA fibres by their fineness and smoothness.

5.4.2.4 Thermal resistance

Measurements of the thermal resistance of the Lots (a measure of warmth) were carried out by Professor Raechel Laing and Assoc. Professor Cheryl Wilson at the University of Otago [95].

A guarded-hotplate [96] was used to determine thermal resistance of the samples. The hotplate is circular, 470 mm in diameter, and generally designed for testing loose fill. However, production of circular duvet samples of this size was not possible—the bonding oven was not wide enough to produce single sheets of bonded knoppy web large enough to use, and using two contiguous semicircles would create thermal leaks and compromise the test. Instead, square samples were produced that could completely cover the central measurement area of the plate and extend out onto the guard. The space around the sample was filled with polyester wadding to ensure the guard was completely covered.

Three samples were prepared for each Lot, consisting of two 400 mm x 400 mm sheets layered within a shell made of Downproof Cotton. The sheets were layered such that the "bottom" sides (as emerging from the DOA airlay) were facing together (so the heavier knops were in the middle of the samples). Additionally, the sheets were layered such that their airlay directions were at right-angles to each other. This was intended to help average out any direction-specific inconsistencies within the roll, that may have occurred during the handling of the unbonded Lots.

The standard test for thermal resistance [96] involves measuring the energy required to maintain the measuring unit at $35 \pm 0.1\,^{\circ}\mathrm{C}$ after equilibrium has been reached. Equilibrium is defined as the rate of energy changing by less than 3% per hour [97]. The thermal resistance is then calculated from the average energy. Because of the modification to the testing procedure, the thermal resistances are indicative only, and can only be compared within the tested samples.

5.4.2.5 Stiffness

An indication of the drape of the knoppy web can be obtained by measuring its stiffness. Test method BS 3356:1961 "Determination of Stiffness of Cloth" [98] was used to measure the stiffness of the knoppy web across the direction of airlay. The measurement apparatus for the test method is a Shirley stiffness tester (fig. 5.21)—in essence, a ledge with lines of known angle $\theta = 41.5°$ scored downwards from the ledge on either side.

A strip of knoppy web approximately 315 mm x 40 mm (the dimensions of a standard ruler) was cut out for each Lot, taken from offcuts of the bonded sheets used for the thermal samples. Each strip was then measured using the following methodology:

1. The strip was placed on the ledge with the ruler placed on top.
2. The zero mark on the ruler was aligned with both the front edge of the strip and the edge of the ledge.
3. Both were then slowly pushed out over the edge of the ledge; the strip would begin to bend downwards under its own weight.
4. The moment the bottom edge of the strip intersected the angled line (fig. 5.22), the distance (that the strip had been pushed out) was read off the ruler.
5. This distance was measured for both ends of the strip and both sides, and from the four values an average length l was determined for the strip.

From the average length l and the angle θ several properties can be calculated, starting with the bending length c—a property that describes how a fabric (or in this case, knoppy web) will bend under its own weight:

$$c = lf(\theta) \text{ cm}, \tag{5.1}$$

where l is in cm and

$$f(\theta) = \left(\frac{\cos \frac{\theta}{2}}{8 \tan \theta} \right)^{\frac{1}{2}}. \tag{5.2}$$

The reason for choosing the particular angle above is because $f(41.5°) = 0.5$.

The second property is the flexural rigidity, G—a measure of stiffness associated with handle:

$$G = \frac{w}{10^4} c^3 \text{ g cm}, \tag{5.3}$$

where w is the weight of the knoppy web in g m^{-2}.

Finally, the bending modulus q can be regarded as the "intrinsic stiffness," and is independent of the dimensions of the tested strip:

$$q = \frac{12G}{g^3} \text{ g cm}^{-2}, \tag{5.4}$$

where g is the thickness of the knoppy web in cm.

Interpretation When interpreting these numbers, the difference between the flexural rigidity and the bending modulus must be carefully understood. G contains a dependency on the thickness of the sample, whereas q is independent of thickness, and can be considered an inherent property of a material.

For comparing products that are intended to be nearly identical, q should be used. Variations in weight and thickness are caused by the inherent variability of the production process; comparing q values remove these variations and provides a better comparison of the effects of the blend composition changes. This is a slight simplification—the differing bulk of the PLA-varied knoppy web blends also contributes to their variation, and is factored out, but we assume this to be a negligible effect.

Figure 5.21: A "Shirley Stiffness Tester" designed for testing webs

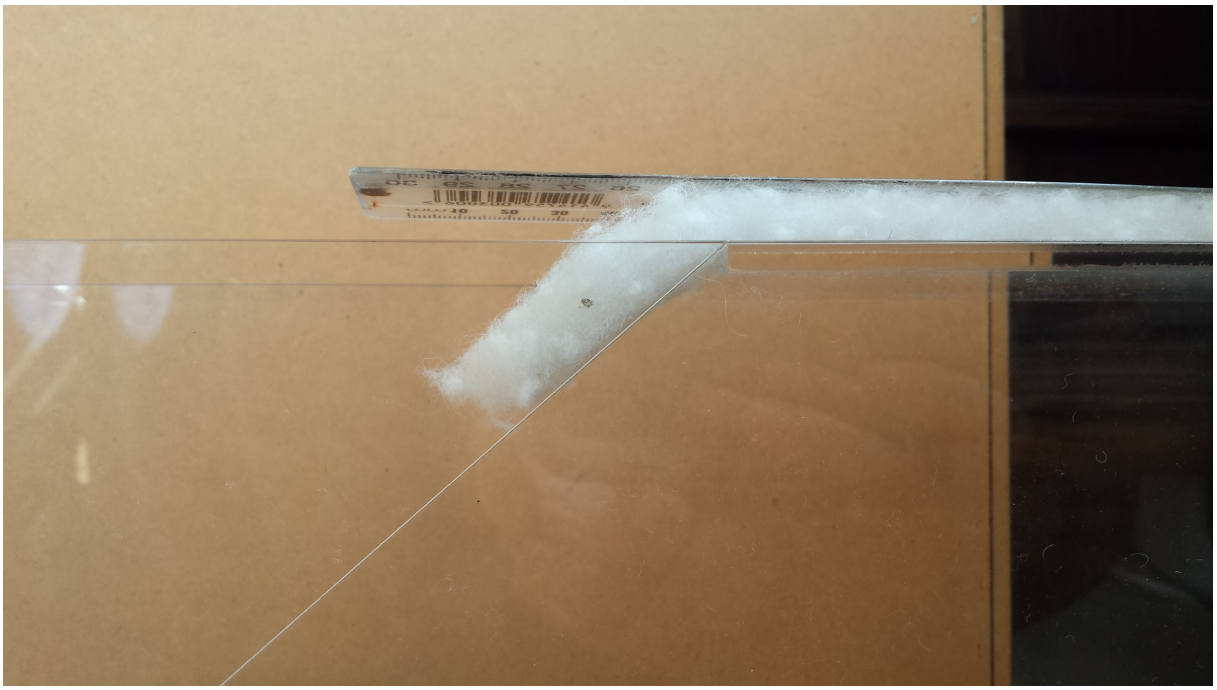


Figure 5.22: Measuring with the Shirley stiffness tester

For comparing the drape and handle of different products that are provided "as-is," G should be used. The ranking of q values is non-intuitive, because it is possible for a thicker, stiffer sample to have a lower q. G values more closely match how a user would qualitatively rank samples.

In this experiment, the independent parameter is the blend composition; the goal of this particular test was to examine the effect of the blend composition on the stiffness. Equation (5.4) has an inverse cubic dependence on the sample thickness, so small variations in thickness across the sample can have a large effect on the calculated q. Unfortunately, due to the fact that the Lots could not be bonded directly after airlay, the thicknesses of the bonded sheets (after rolling, unrolling, cutting, and transporting the unbonded knoppy web) were not sufficiently consistent. Therefore, only one strip was tested from each Lot, and they were carefully selected from the available bonded material to have uniform thickness. This means that the calculated values of c and G are not particularly meaningful, but they are not necessary within the context of this test.

5.4.2.6 Wash tests

Wash tests are generally performed on either loose filling (such as loose knops) or the final product (e.g. a duvet); the small-scale knoppy web trials conducted fall between these two categories. The standard wash test for loose knops [99] and the associated bulk test [87] were modified for testing knoppy web:

1. A circular knife was used to cut 188 mm diameter discs from bonded knoppy web sheets.
2. The discs were stacked inside the standard bulk cylinder to a height of 160 mm to 180 mm (about 6–8 discs).
3. The standard bulk test was performed as per TM 272 [87].
4. The discs were placed inside a cotton carry bag (as for loose fill), with non-woven fabric separating individual discs to avoid aggregation.
5. The standard wash test was performed as per TM 274 [99] (3 x 7A washes).
6. The discs were dried in a regain drier at 82.5°C for 0.65 hours.
7. A post-wash bulk test was performed as above.

Because of the novelty of the knoppy web wash test, wash test grades were not assigned (the original grading scale for loose knops was meaningless for knoppy web).

5.4.2.7 Force-displacement curves

A variety of bonded samples from the various Lots (obtained during the preparation of samples for the above tests) were collected and placed into storage. The samples were left in the open for 9 months, and then placed in a box for 17 months. This aging process provides a more realistic view of the expected compression behaviour of the Lots as experienced by consumers. Force-displacement curves were then measured after conditioning all samples for a minimum of 24 hours under standard conditions (20 °C, 65% relative humidity).

The force-displacement curves were measured with an Instron 4204 [91] fitted with a 100 kg load cell and a 100 cm^2 circular foot (fig. 5.23). The foot was lowered onto each knoppy web sample at a rate of 50 mm min^{-1} until a height of 5 mm was reached, and then raised at the same rate back to the starting height.

The thicknesses of the samples were defined by the height at which the measured force was no longer negative. The force-displacement curves were normalised by dividing the displacement by the thickness of the sample. Figures were prepared showing force, force-per-thickness, and force-per-weight (using averaged weights of the lots, not specific to each measured sample).

5.4.3 Results

5.4.3.1 Qualitative assessment

Prior to airlay, the different knoppy web blends varied mainly in bulk: the blends with more total PLA appeared to be bulkier. This makes sense, because the PLA fibres are finer than the wool fibres and so for equivalent weights there will tend to be a greater number of PLA fibres. However, the blend for Lot 6 appeared to be even bulkier before airlay than Lot 4 (which had the highest PLA content). This makes sense given another observation: the knops within Lot 6 appeared to have almost been "carded away" into the web, as shown in fig. 5.24. While other Lots had a large fraction of fibres stored in the relatively dense knops, Lot 6 had more opened fibre within the web (even compared to Lot 2 which had the same percentage of web in its blend).

These bulk observations seemed to carry over into the thickness of the knoppy web coming off the airlay. Unfortunately, some of this bulk was lost when the knoppy web was rolled up and stored. Without a continuous bonding oven to bond the PLA in-place, the bulk was lost by compression within the rolls under their own weight.

The pre-bonding handling of the knoppy web affected other properties as well. The initial few sheets cut from the first Lot to be bonded were transported to the oven draped over an arm. Post-bonding, it became apparent that this had caused the knoppy web to begin to pull apart under its own weight, leaving "holes"—thin patches that were visible with a backlight and that would compromise the thermal properties of the knoppy web.

Subsequent sheets were transported on cardboard with minimal disturbance, and that particular pattern of "holes" were no longer seen. However, the sheets did retain a degree of patchiness, pointing to sub-optimal blending of the knops through the web. The only blending that the knoppy web blends underwent (Lot 6 aside) was to pass through the opener; while an industrial-scale run may see better results, a more effective knop-web blending process would be beneficial (though of course effectiveness needs to be balanced against efficiency). Conversely, the Lot 6 knoppy web appeared to be better blended, but the fact that the knops were close to non-existent means that the blending process had progressed too far. The Lot 6 blend looked strikingly similar to a carded sliver with "nepping," and the unique value of having a "knoppy web" was arguably lost.

Compared to the unbonded samples (which as mentioned above were sensitive to post-airlay processing), all bonded knoppy webs had excellent physical integrity, and would be easy to handle for manufacture of end products such as duvets.

After bonding the Lots with higher PLA fractions had less drape and were much stiffer. In particular, Lots 3 and 4 were deemed unsuitable for overbody usage. Lot 2 was only marginally acceptable from a stiffness point of view. The Lot 6 sheets had the softest feel, suggesting that for overbody products the additional carding step might be necessary (though to a lesser degree so as to better preserve the knop structure). The stiffer Lots did appear to have better recovery after compression by hand, and may be better-suited for underbody products.

5.4.3.2 Indicative shrinkage during bonding

Table 5.6 gives the indicative shrinkage by area observed while bonding sheets for the thermal samples. There appears to be some correlation to the wool fraction in the Lots (table 5.5)—the shrinkage increases as the wool fraction decreases. This is consistent with the idea that the PLA component within the knoppy web shrinks when heated, and therefore the Lots with a larger overall percentage of PLA will shrink more. The largest increase in shrinkage was observed for varying the PLA content of the web. While this did create the largest variation in total PLA content within the experiment, its increasing presence in the web was also an important factor.

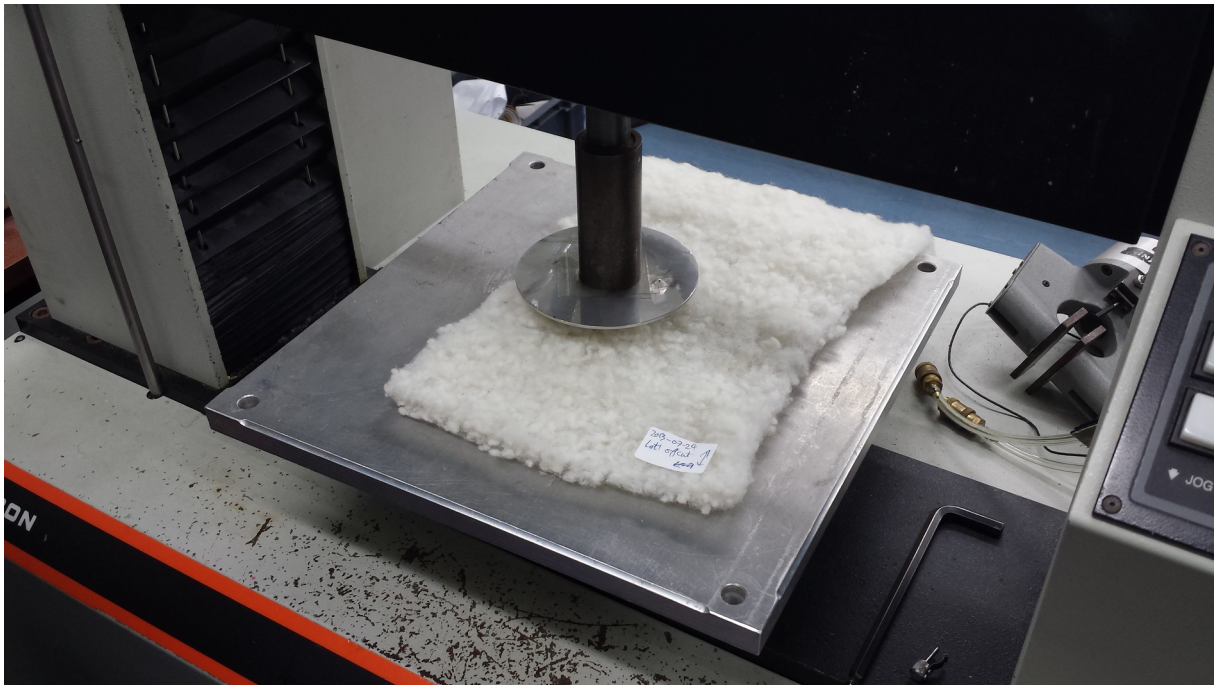


Figure 5.23: The compression apparatus for measuring force-displacement curves of knoppy web

Figure 5.24: Samples of Lot 6 (top) and Lot 2 (bottom), showing how the knops have been removed by the extra carding step.

Table 5.6: Indicative shrinkage by area during bonding.

Lot	Knop:Web	Indicative shrinkage
1	90:10	14.5%
2	80:20	14.5%
3	70:30	16.9%

(a) Knop:web variation

Lot	PLA:Wool	Indicative shrinkage
5	25:75	8.8%
2	50:50	14.5%
4	75:25	19.0%

(b) Web blend variation

Lot	Carded?	Indicative shrinkage
2	No	14.5%
6	Yes	13.5%

(c) Carding

The effect of bonding on the thickness of the knoppy webs was not measured and not considered within the indicative shrinkage results. This is due to the bulk loss mentioned in previous Sections. The heat within the bonding oven will likely have caused thickness to decrease as it did area, but counteracting that will have been the heat-induced relaxation of the knoppy web from its compressed state. Therefore, no meaningful results would be obtained from comparing thickness measurements of samples from rolled-up unbonded knoppy web pre- and post-bonding.

Bonding shrinkage is a factor that will need to be considered when setting up a production line. For example, if a product requires a 2 m-wide roll of bonded knoppy web, then both the DOA airlay and the continuous bonding oven would need to accommodate up to 2.5 m-wide unbonded knoppy web (for a blend similar to Lot 4). That said, the larger product width itself may actually inhibit area shrinkage, due to increased friction between the knoppy web and the bonding oven feed sheet.

5.4.3.3 SEM

For a macroscopic product such as knoppy web, it is difficult to give a representative overview of microscopic-scale images within a summary document such as this. Nevertheless, some interesting features have been identified which are outlined below.

Figure 5.25 shows a view of the web component of a sample from Lot 2. This shows what is to be expected of an effective 50:50 web—good blending of the wool and PLA fibres, and good PLA-PLA bonds. However this was not always the case: general observation of the samples under the SEM tended to show wool-rich regions and PLA-rich regions, indicating that there was sub-optimal microblending. This was more obvious in the knop samples (where the PLA content was intrinsically lower); for example, fig. 5.26 shows a view of a knop from Lot 4 where no PLA fibres are visible.

Figure 5.27 exemplifies another feature that was discovered: there were sections within both the knops and web where several PLA fibres would become completely fused together, or create complex multi-fibre bonds. The simplest explanation is that the crimp in the stock PLA fibre was not entirely removed during processing, and groups of PLA fibres survived through to the final airlay. This would also explain the tendency for PLA-rich regions to form. The PLA fibre is finer and much more "slippery" than wool, so it makes sense that machinery designed for processing wool might not be as efficient at processing and separating PLA.

Figure 5.28 shows several good long PLA-PLA bonds within the Lot 1 web. It also clearly exhibits the effect of production-time handling on the knoppy web structure: there is a large longitudinal tear of a PLA-PLA bond, believed to be caused by post-bonding manipulation. The sheets being individually bonded in the oven were manually handled while the bonds were still soft, and in some cases the edges of the sheets required uncurling (having been rolled up by the internal airflow). From an industrial standpoint, this indicates that the knoppy web needs adequate cooling after passing through the bonding oven before being rolled up or otherwise handled.

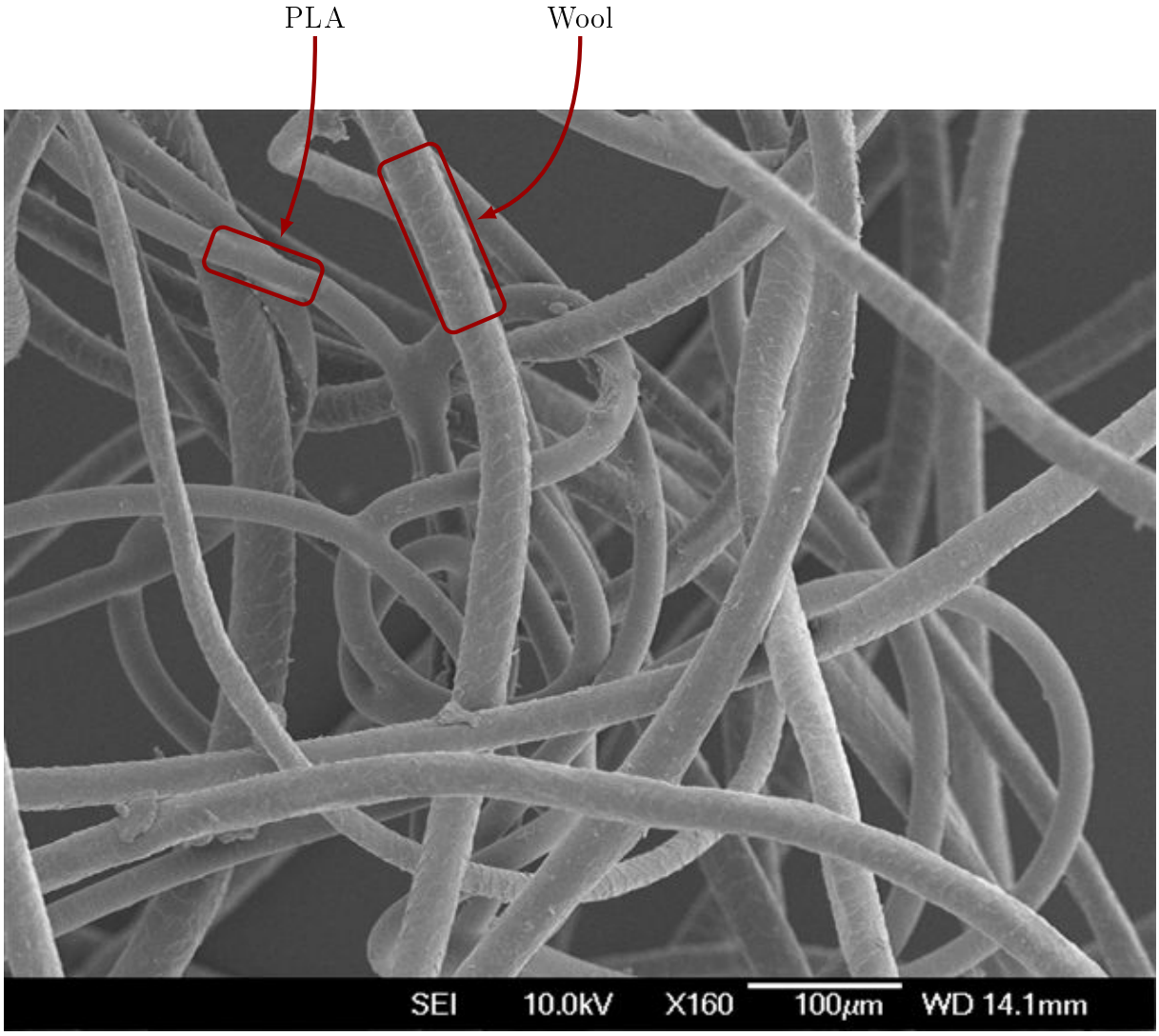


Figure 5.25: SEM image of Lot 2 web

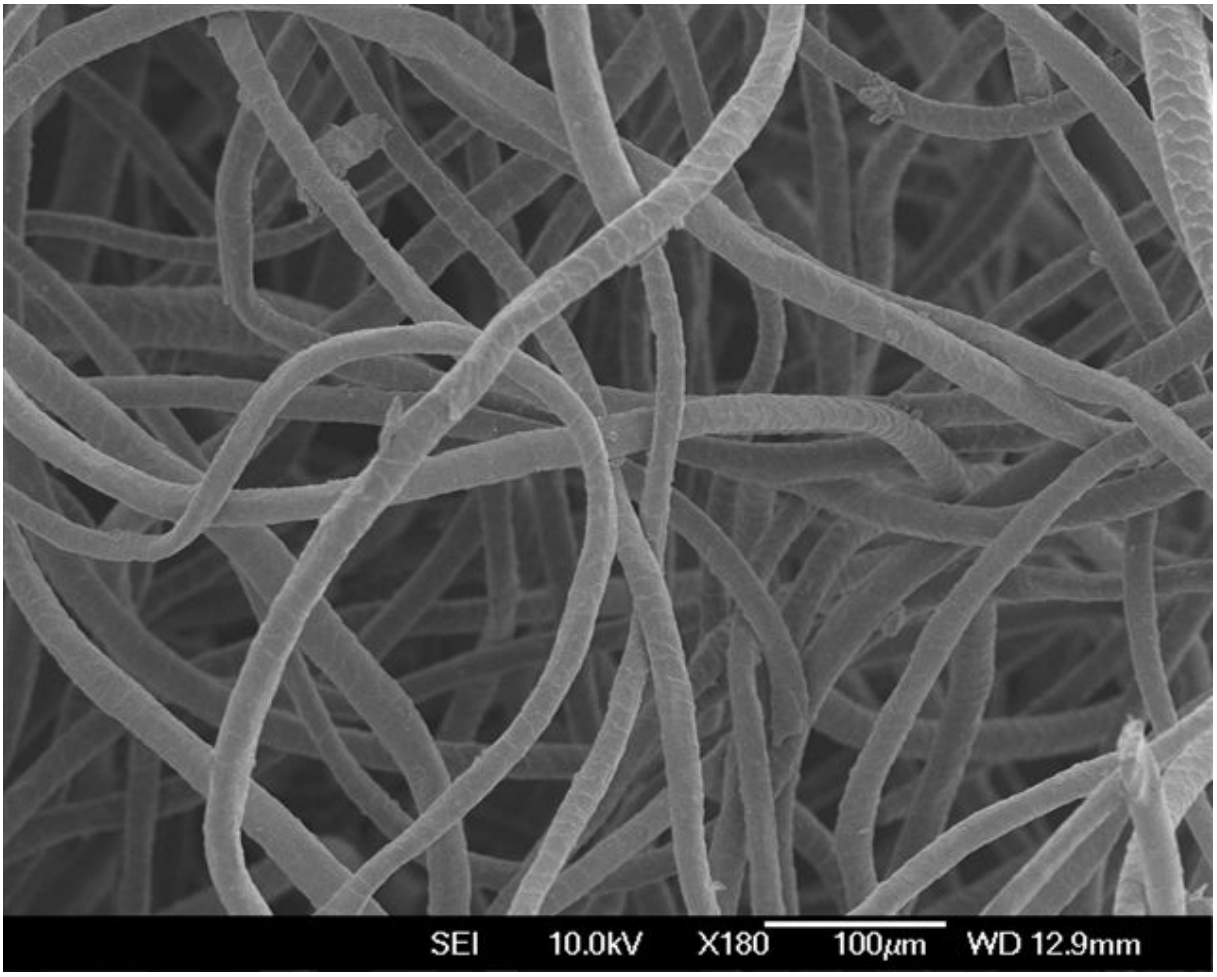


Figure 5.26: SEM image of Lot 4 knop

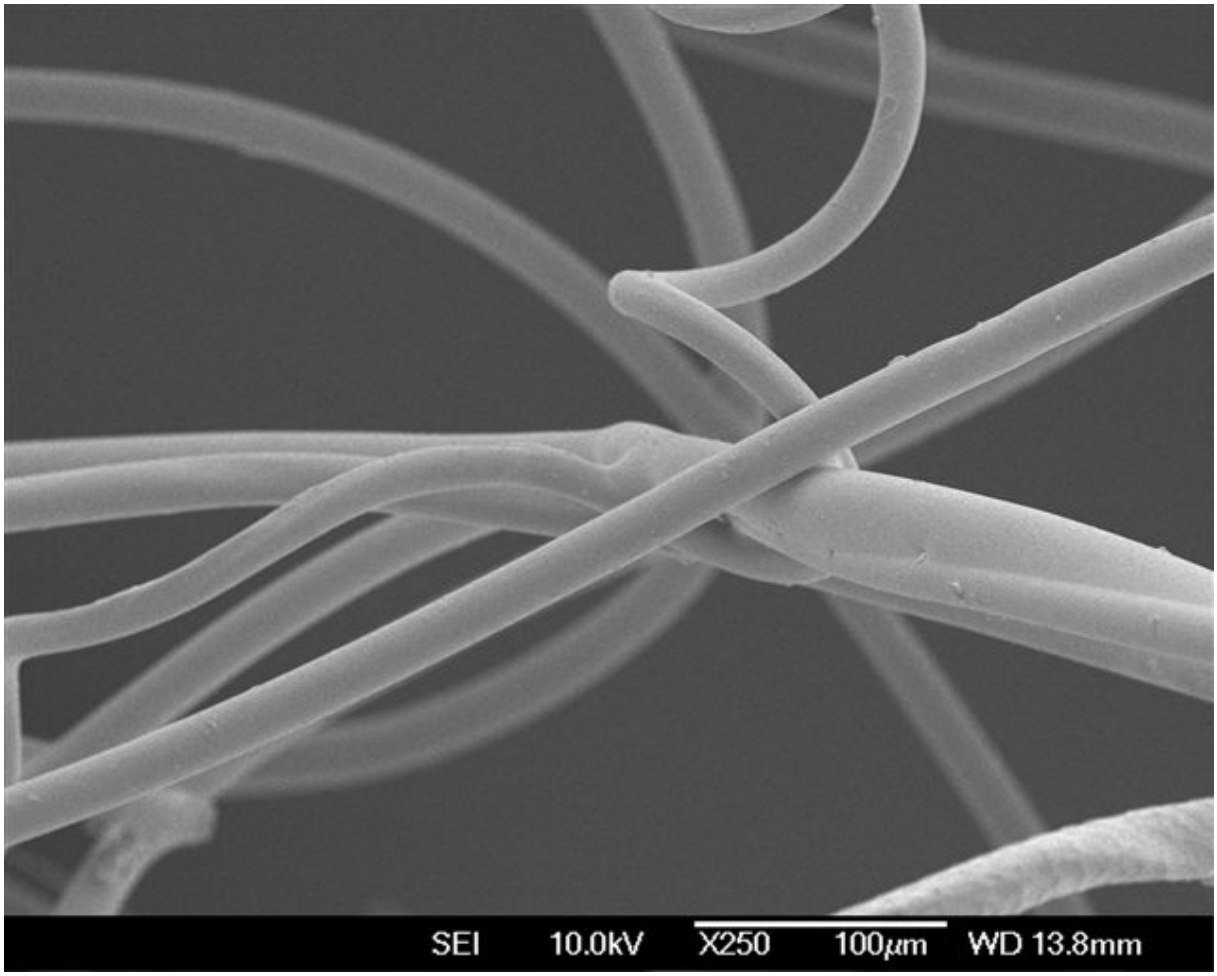


Figure 5.27: SEM image of Lot 4 web

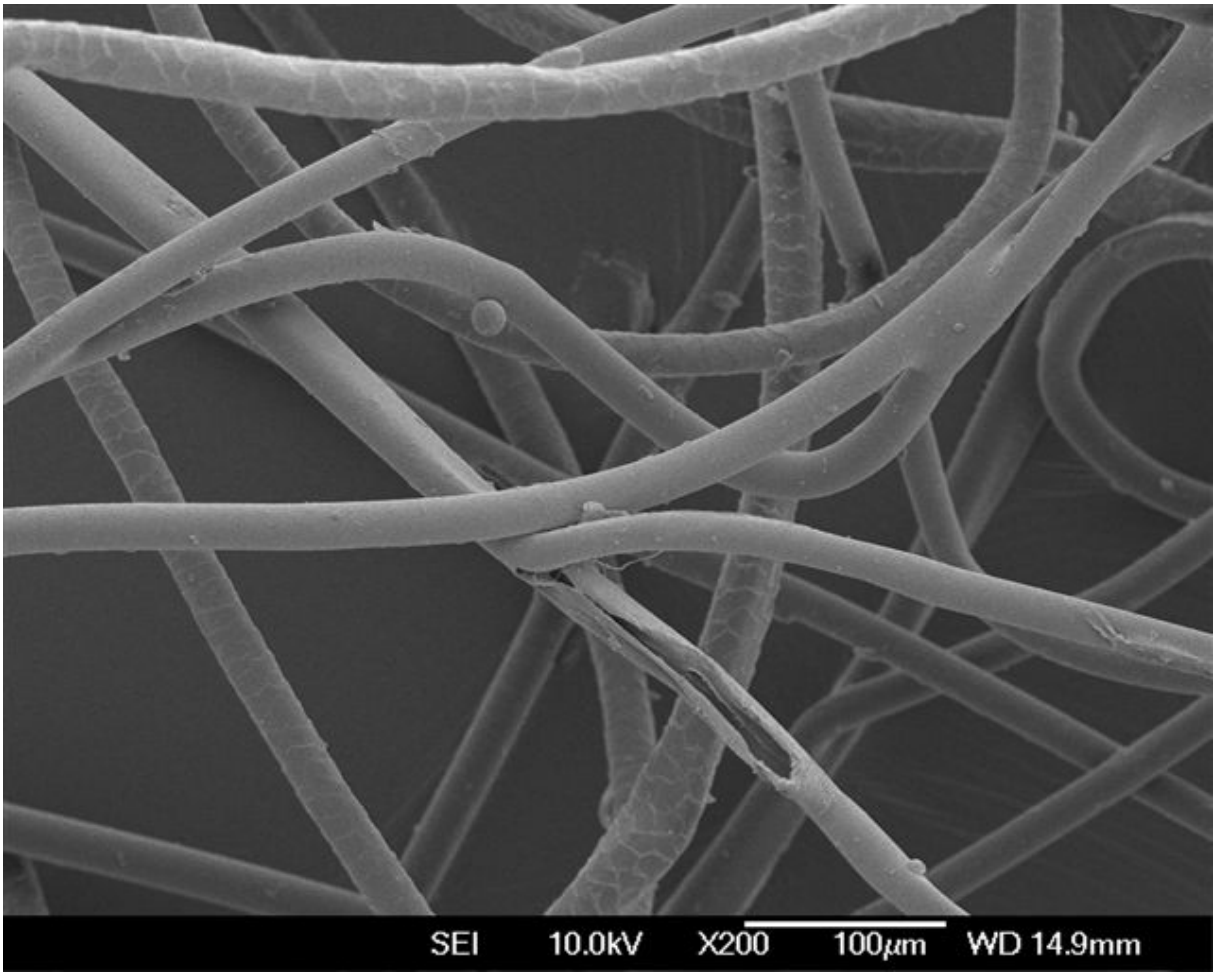


Figure 5.28: SEM image of Lot 1 web

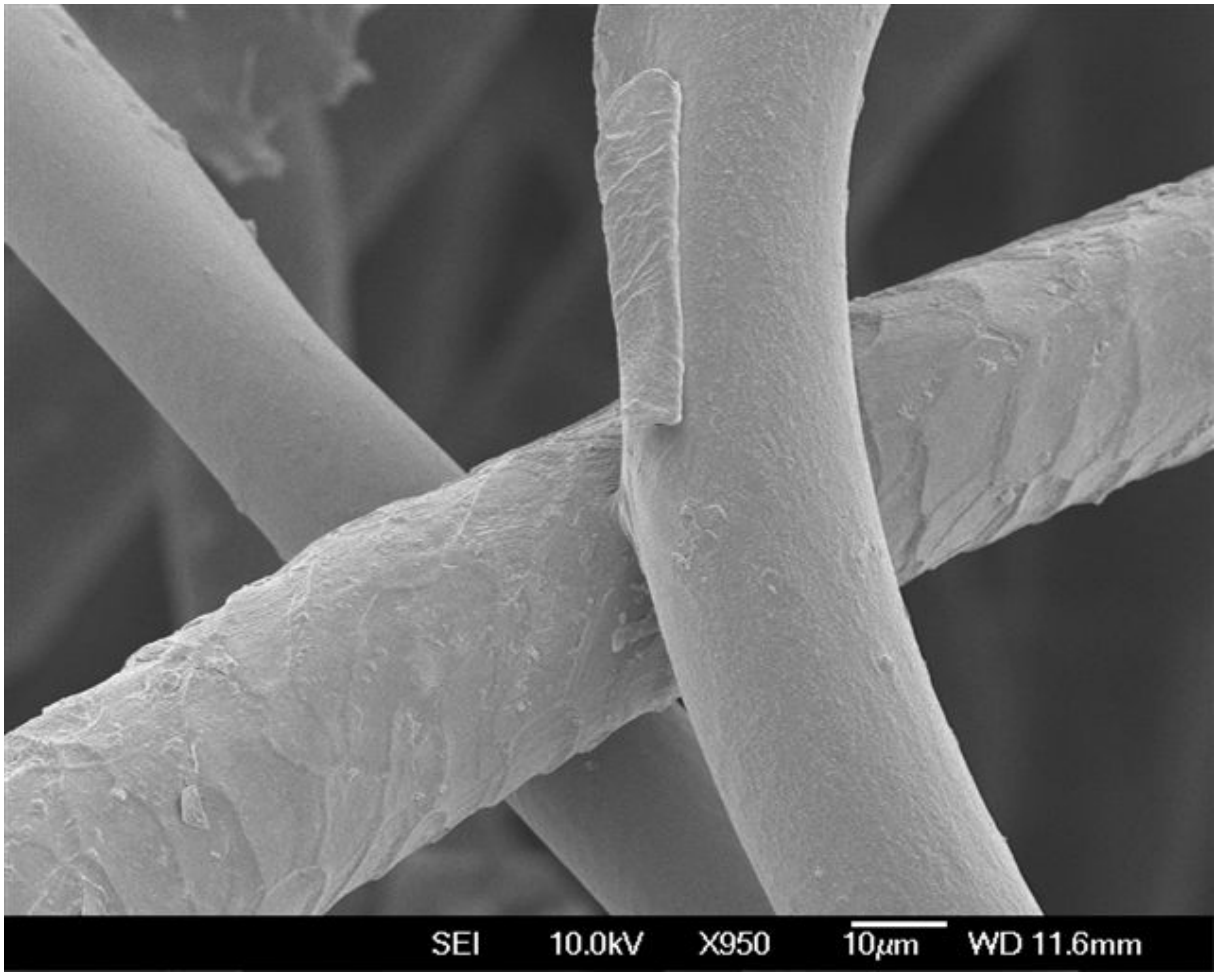


Figure 5.29: SEM image of Lot 3 knop

Figure 5.29 shows a view of a knop from Lot 3 with a detached wool cuticle stuck to the PLA fibre. Such detritus was commonly observed in the SEM micrographs, and suggests that the adhesion coefficient between PLA and wool cuticles is in some cases strong enough to exceed that between the cuticle and the wool core. Far less common is the other feature in the figure: a small but intact PLA-wool bond. These observations indicate that PLA-wool bonds cannot be relied on for structural integrity of the knoppy web; even when they do form, subsequent processing (into final products) and usage (compression, washing) appears to cause the PLA-wool bonds to fracture.

5.4.3.4 Thermal resistance

Table 5.7 shows the inherent thermal resistances, warmth to mass and warmth to thickness ratios of the Lots. The coefficients of variance for the inherent resistances were all 4% or less, indicating that the prepared samples were relatively homogeneous.

The key results of the thermal resistance testing were as follows [95]:

1. The knop:web ratio affected the thermal resistance of the samples, but the PLA:wool ratio in the web did not.
2. Lots 3 and 6 were generally the "warmest" per gram and per unit thickness.
3. Little difference in warmth was observed between Lots 3 and 6.
4. Lots 1, 2, 4 and 5 were least warm on a warmth:mass comparison, and Lot 1 was intermediate between the most and least warm groups when warmth to thickness ratios were examined.

5.4.3.5 Stiffness

Table 5.8a gives the measured properties of the Lot strips tested. The weight values for the various Lots were determined during preparation of the thermal samples (from the measured weight of the layers and their known dimensions). The thicknesses were estimated with a measuring tape. Table 5.8b gives the calculated values for the bending length c, flexural rigidity G and bending modulus q for each Lot.

Table 5.9a shows that increasing the fraction of knops in the knoppy web decreases the bending modulus, improving drape. Furthermore, the relationship appears to be linear over the examined range (fig. 5.30(a)). This gives support to hypothesis 2, which proposed that the decrease in fibres aligned with the plane of the knoppy web would improve drape:

- If we assume that the web's fibre ratio can be represented by a single fibre type of some intermediate diameter, then the number of web fibres in the knoppy web will be directly proportional to its web fraction.
- Assuming that the knops store negligible bending strain, the bending strain stored in each fibre will be equal to the total bending strain divided by the number of fibres, and therefore inversely proportional to the web fraction. This matches the observed relationship.

Table 5.9b shows that there is a clear correlation between the fraction of PLA in the web and the bending modulus. The relationship appears to be near-cubic (fig. 5.30(b)). Increasing the fraction of PLA within the knoppy web increases the number of PLA-to-PLA contact points; when bonded, this gives the knoppy web a stiffer structure. This result agrees with the qualitative assessment of the bonded sheets (as outlined in section 5.4.3.1).

It is interesting to note that Lot 6 has a considerably lower bending modulus than Lot 2 (table 5.9c), despite them having identical blends. The additional carding process prior to airlay clearly affected the underlying structure (as was evident from visual inspection)—the knops in Lot 6 appeared to have almost been "carded away" into the web. This would have increased the fibre density of the web, but

Table 5.7: Inherent thermal resistance, warmth to mass and warmth to thickness ratios of Lots [95]. (μ = mean, σ = standard deviation, c_v = coefficient of variation)

Lot	Thermal resistance ($m^2\,K\,W^{-1}$)			Warmth to mass ratio ($m^2\,K\,W^{-1}\,g^{-1}$) $\times 10^{-4}$			Warmth to thickness ratio ($m^2\,K\,W^{-1}\,mm^{-1}$) $\times 10^{-2}$		
	μ	σ	c_v %	μ	σ	c_v %	μ	σ	c_v %
1	1.181	0.013	1	10.21	0.68	7	2.11	0.08	4
2	1.187	0.045	4	10.14	0.33	3	2.04	0.11	5
3	0.862	0.019	2	11.60	0.60	5	2.19	0.09	4
4	1.173	0.019	2	9.69	0.27	3	1.93	0.02	1
5	1.208	0.011	1	9.98	0.31	3	2.01	0.10	5
6	1.121	0.030	3	12.50	0.20	2	2.25	0.04	2

Table 5.8: Stiffness properties of the various Lots across the airlay

Lot	Weight ($\pm 3\,g\,m^{-2}$)	Est. thickness (± 0.25 cm)	Avg. length (± 0.1 cm)
1	410	3.0	12.3
2	404	3.5	21.8
3	219	2.0	18.1
4	413	2.0	17.5
5	402	3.3	10.4
6	275	2.5	14.3

(a) Measured properties

c (cm)	G (g cm)	q ($g\,cm^{-2}$)
6.15 ± 0.05	9.5 ± 0.3	4.2 ± 1.2
10.90 ± 0.05	52.3 ± 1.1	14.6 ± 3.4
9.05 ± 0.05	16.2 ± 0.5	24.3 ± 9.9
8.75 ± 0.05	27.7 ± 0.7	41.5 ± 16.6
5.20 ± 0.05	5.7 ± 0.2	2.0 ± 0.5
7.15 ± 0.05	10.1 ± 0.3	7.7 ± 2.6

(b) Stiffness parameters

Table 5.9: Comparison of stiffness properties

Lot	Knop:Web	q ($g\,cm^{-2}$)
1	90:10	4.2 ± 1.2
2	80:20	14.6 ± 3.4
3	70:30	24.3 ± 9.9

(a) Knop:web variation

Lot	PLA:Wool	q ($g\,cm^{-2}$)
5	25:75	2.0 ± 0.5
2	50:50	14.6 ± 3.4
4	75:25	41.5 ± 16.6

(b) Web blend variation

Lot	Carded?	q ($g\,cm^{-2}$)
2	No	14.6 ± 3.4
6	Yes	7.7 ± 2.6

(c) Carding

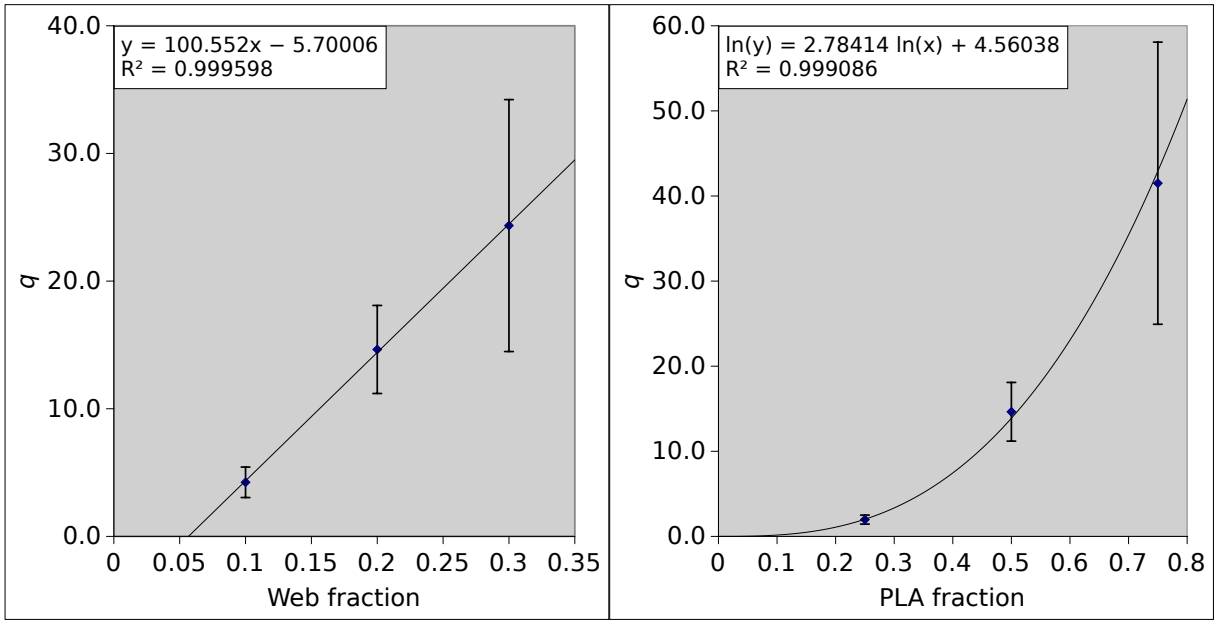


Figure 5.30: Effect on the bending modulus q of (a) the knop:web ratio and (b) the PLA:wool ratio in the web.

would also have lowered the effective percentage of PLA in the web due to the higher fraction of wool in the knops. Thus, the additional carding process lowers the number of PLA-to-PLA contact points available for bonding, making the knoppy web less stiff.

5.4.3.6 Wash tests

Wash tests were performed on all of the Lots. Additionally, wash test results for both untreated and shrink-resist-treated wool, and for pure polyester, were conducted to provide a comparison to the status quo.

All samples lost bulk across the three wash tests, due both to decreasing thickness and area shrinkage. This shrinkage was generally non-uniform between discs and between directions within each disc. The discs were qualitatively observed during and after the wash tests. For all samples, the following changes appeared incrementally across the three washes:

- The untreated wool discs underwent considerable felting and shrinkage (estimated around 60% by area), and some distortion (fig. 5.31).
- The shrink-resist-treated wool discs were drastically distorted, and there was considerable fibre migration within them creating thick regions as well as complete holes (fig. 5.32).
- The polyester discs showed minimal distortion and had the least area shrinkage of all tested samples (estimated around 15% by area) (fig. 5.33).
- Area shrinkage was approximately the same for all Lots (estimated about 25% by area) (figs. 5.34 to 5.36).

Qualitatively, all knoppy web samples outperformed the pure wool samples. However, as is clear from the various figures, they were not without change. In all Lots, the knops appeared to get bigger, and the web less uniform; this indicates that the web fibre was being worked into the knops.

The post-wash bulk and bulk loss figures were obtained from the same bulk testing procedure as the pre-wash bulk. However, these do not take the area shrinkage into account, which affects the bulk test in two ways:

- The bulk calculation assumes that the diameters of all discs are equal to the 190mm inner diameter of the testing cylinder; this is not the case post-wash.
- Because the disc diameters were smaller than the testing cylinder diameter, the discs were laterally unconstrained when placed under compression and could expand sideways, resulting in slightly more compression and a smaller height measurement than if a testing cylinder of the same diameter was used.

These have opposing effects on the measured bulk—the assumed diameter inflates the bulk, and the lack of lateral constraint decreases it. However, the difference in measured height is negligible compared to both the magnitude of the area loss, and the uncertainty in the true area. Therefore, adjustments were made to the measured bulks to account solely for the area loss. The adjusted post-wash bulk and adjusted loss figures were calculated based on an approximate average disc diameter of 130 mm for the untreated wool, 175 mm for the polyester, and 165 mm for all Lots. Unless explicitly stated, bulks and losses referred to below are the adjusted bulks and losses. These should be taken as approximate lower bounds on the post-wash bulks.

It should be noted that area shrinkage is not necessarily important to consider; in washable finished products such as duvets, the knoppy web would likely be quilted into place, which would inhibit (though not prevent) any area shrinkage.

(a) Pre-wash (b) Post-wash

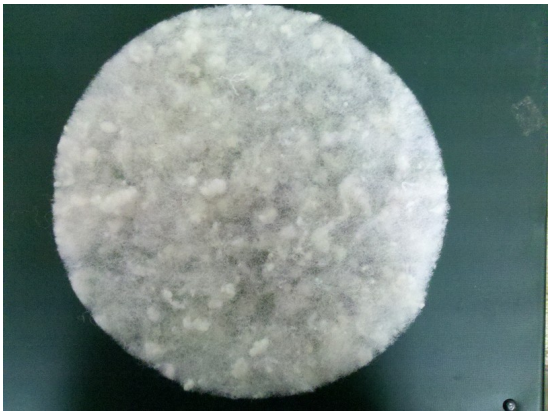
Figure 5.31: Untreated wool

(a) Pre-wash (b) Post-wash

Figure 5.32: Shrink-resist-treated wool

(a) Pre-wash (b) Post-wash

Figure 5.33: Polyester




(a) Pre-wash (b) Post-wash

Figure 5.34: Lot 3


(a) Pre-wash (b) Post-wash

Figure 5.35: Lot 5


(a) Lot 2 post-wash (b) Lot 6 post-wash

Figure 5.36: Other Lots

Table 5.10: Wash test results. * = the Lots were wash tested after 22 months in storage.

Sample	Pre-wash bulk (cc/g)	Post-wash bulk (cc/g)	Bulk loss	Adj. post-wash bulk (cc/g)	Adj. loss
Untreated wool	61.5	51.4	16.4%	24.1	60.8%
Shrink-resist wool	74.6	Could not measure		N/A	
Polyester	142.9	106.3	25.6%	90.2	36.9%
Lot 1*	56.0	56.2	-0.4%	42.4	24.3%
Lot 2	66.5	54.4	18.2%	41.0	38.3%
Lot 3	79.6	58.8	26.1%	44.3	44.3%
Lot 4*	63.8	55.6	12.9%	41.9	34.3%
Lot 5	65.8	57.7	12.3%	43.5	33.9%
Lot 6	92.9	65.3	29.7%	49.2	47.0%

Table 5.10 shows the wash test results. The quantitative measurements back up the earlier qualitative observations: the knoppy web samples greatly outperformed the pure wool samples. The untreated wool lost 60.9% of its bulk, and is a textbook example of why pure wool products are not machine-washable. The shrink-resist-treated wool was distorted to the point that its bulk could not be measured post-wash. It is believed that the shrink-resist treatment prevented global felting, and fibre migration within the samples caused the localised felting and distortion.

The polyester showed almost no area shrinkage, but had a bulk loss comparable to the various Lots. The former makes sense because polyester is a smooth fibre that felts much less readily than wool. However, the bulk loss of the polyester was in reality higher than the various Lots. The result was skewed by the fact that the polyester pre-bulk measurement height was considerably lower than its zero-pressure standing height, much more so than for any other sample (polyester is weak under compression).

Table 5.11a shows the effect of altering the knop:web ratio—increasing the percentage of web in the knoppy web increased the bulk, but also increased the bulk loss. The latter could have been caused both by the lower compression resistance of the web relative to the knops, and an increase in availability of web fibres to felt.

Table 5.11b shows the effect of altering the PLA:wool ratio. At first glance there does not appear to be any meaningful correlation between the percentage of PLA in the web and the bulk loss. However, due to time constraints the Lot 2 discs were cut (and wash testing started) within an hour of being bonded; by comparison, all other Lots were bonded well before being cut and wash tested. The short turnaround for Lot 2 could have compromised its PLA bonding, resulting in a higher bulk loss. If we assume that the bulk loss for Lot 2 is higher than it should be by about 4%, then table 5.11b indicates that increasing the percentage of PLA in the web has no effect on the bulk loss. This assumption is not unreasonable, and does not affect the conclusions drawn from the other tables.

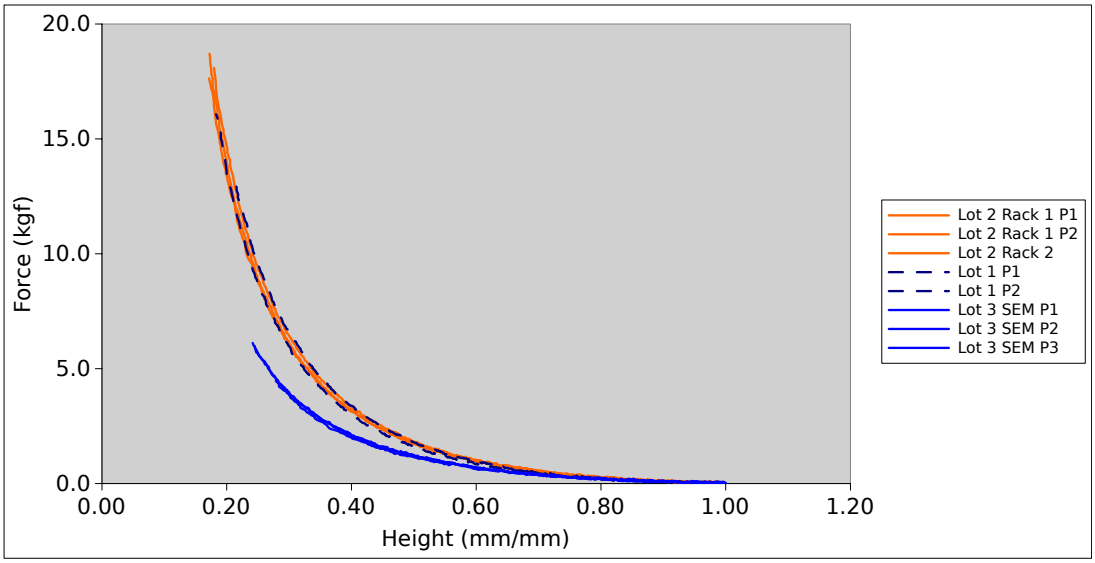
Table 5.11c shows the effect of carding the knoppy web, and along with the table 5.10 reveals some interesting results:

- Lot 6 had higher pre-wash and post-wash bulks than all other Lots.
- Lot 6 had a higher bulk loss than all other Lots.

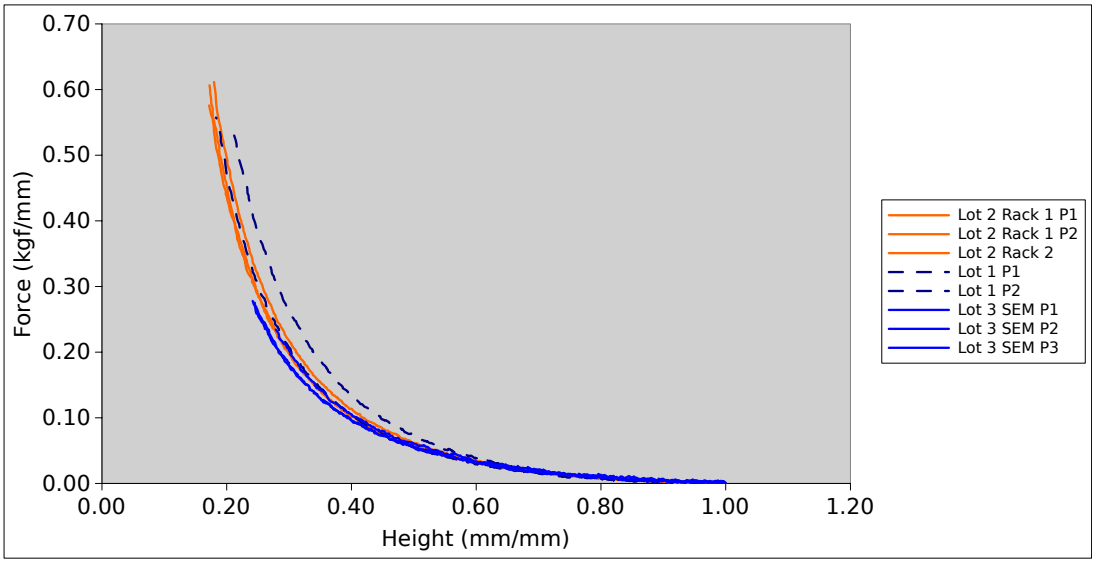
This strongly suggests that while some carding is beneficial to the knoppy web to increase bulk, the level of carding present in Lot 6 compromised the ability of the knoppy web to withstand the harsh wash cycle testing. Lot 6 was composed primarily of web (its knops having been "carded out"), which has a much lower compression resistance than the knops.

5.4.3.7 Force-displacement curves

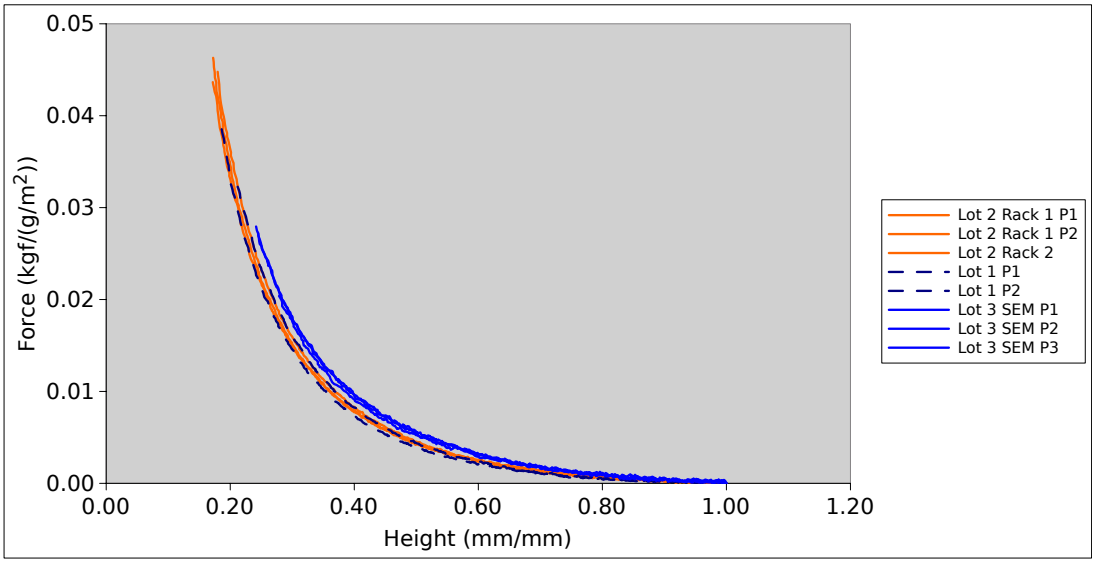
Figure 5.37 shows the effect of varying the knop:web ratio. After factoring out differences in thickness (fig. 5.37b), it is possible to see that decreasing the fraction of knops in the knoppy web does decrease



(a) Force



(b) Force per thickness



(c) Force per weight

Figure 5.37: Effect on force-displacement curves of varying the knop:web ratio.

Table 5.11: Wash test comparisons

Lot	Knop:Web	Adj. loss
1	90:10	24.3%
2	80:20	38.3%
3	70:30	44.3%

(a) Knop:web ratio

Lot	PLA:Wool	Adj. loss
5	25:75	33.9%
2	50:50	38.3%
4	75:35	34.3%

(b) Web blend ratio

Lot	Carded?	Adj. loss
2	No	38.3%
6	Yes	47.0%

(c) Carding

the developed strain energy, and thus decreases its resilience as expected, although not greatly over the range considered (90% knops for Lot 1, 70% knops for Lot 3). However, factoring out weight (fig. 5.37c) slightly reverses the trend, with Lot 3 becoming the most intrinsically resilient.

Figure 5.38 shows the force-displacement curves for the Lots where the wool:PLA ratio in the web was varied. The Lot 5 #2 sample is considered an outlier, due to the appreciable deviation both between the two measured points, and compared to the other Lot 5 sample. Looking at the remaining samples, Lot 5 is possibly slightly less resilient, but otherwise there is no noticeable effect on the compression resilience from changing this ratio. This seems at odds with the stiffness measurements given in table 5.9b. However, stiffness measurements involve a sizeable tension component that is not present in the compression measurements, and we believe that the bonded PLA in the web is appreciably stiffer under tension (where the PLA fibres only incur tensile strain) than compression (where the PLA fibres can incur both compressive strain and bending strain). The most interesting implication of fig. 5.38 is that reducing the PLA content in the web gives a product with a softer feel (per the qualitative assessments in section 5.4.3.1) but that does not require any significant compromise on resilience.

Figure 5.39 shows the effect of additional carding. It is clear that the knops add considerable resilience under compression to the fibrous batt; their removal decreases the force required to reach an equivalent compressive strain by a factor of approximately 2.

5.4.4 Discussion

The results obtained in this experiment have largely followed intuition:

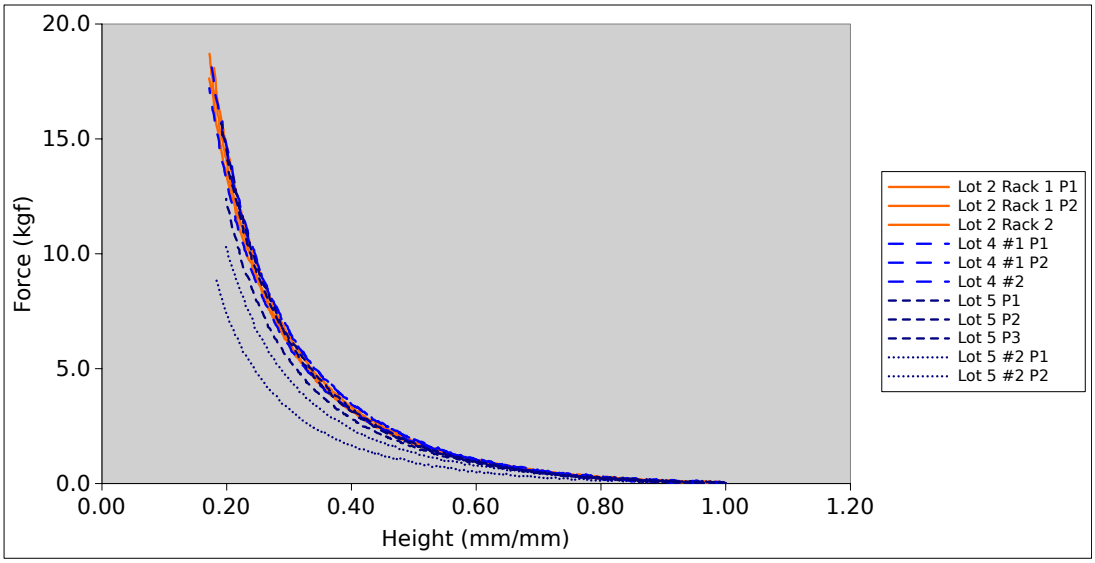
- Blends with a higher web fraction or more PLA content are stiffer.
- Knops improve both drape and washing performance.
- Knops make a significant contribution to resilience.
- Increasing PLA content noticeably decreases qualitative handle and feel.

The application of the results to product development will be covered in a later section. However, there are a few more general observations that can be made about the results.

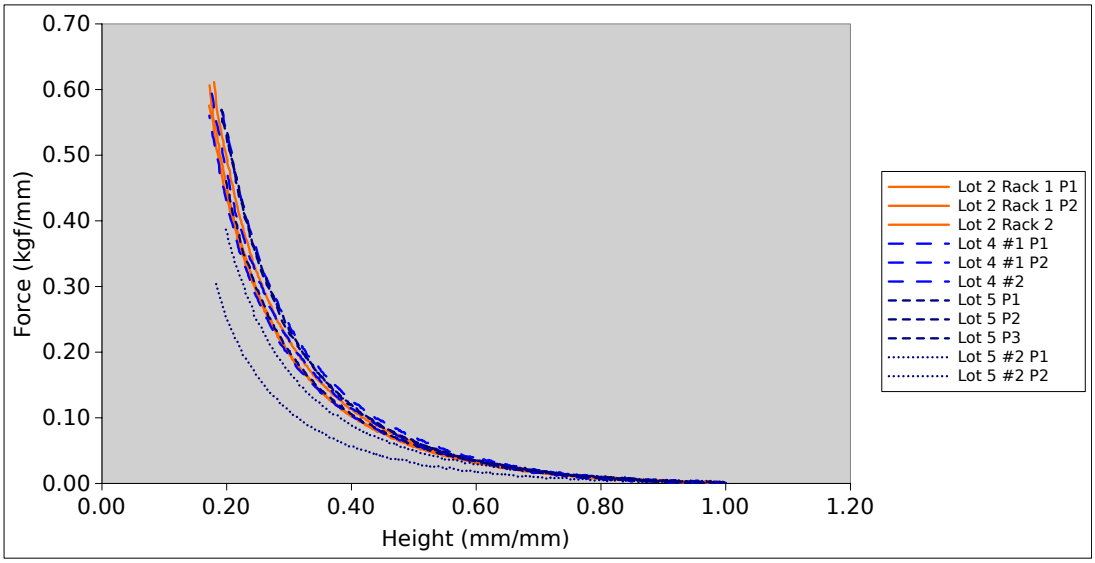
There is a clear tradeoff between warmth and washability. Lots 3 and 6 were generally the warmest samples (table 5.7), but also suffered the highest bulk loss after washing (table 5.10). Both results are due to the higher fraction of web fibre in these Lots:

- The presence of knops in knoppy web creates an uneven distribution of fibre, which can result in "thin" patches occurring. Decreasing the knop fraction (or eliminating it almost entirely, in the case of Lot 6) smooths out the density profile and lessens thermal leakage.
- More web fibre means a higher felting ability, which causes the bulk loss after washing.

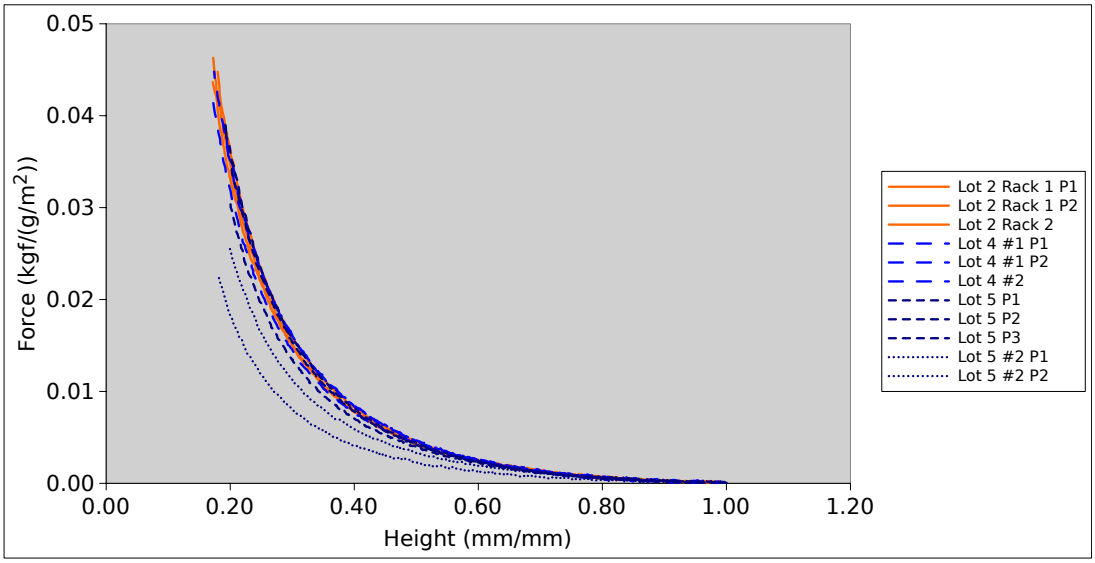
Therefore, when designing a knoppy web blend, consideration must be given to the likely end uses of the product. For a product where warmth is a key factor, washability may need to be sacrificed (by



(a) Force

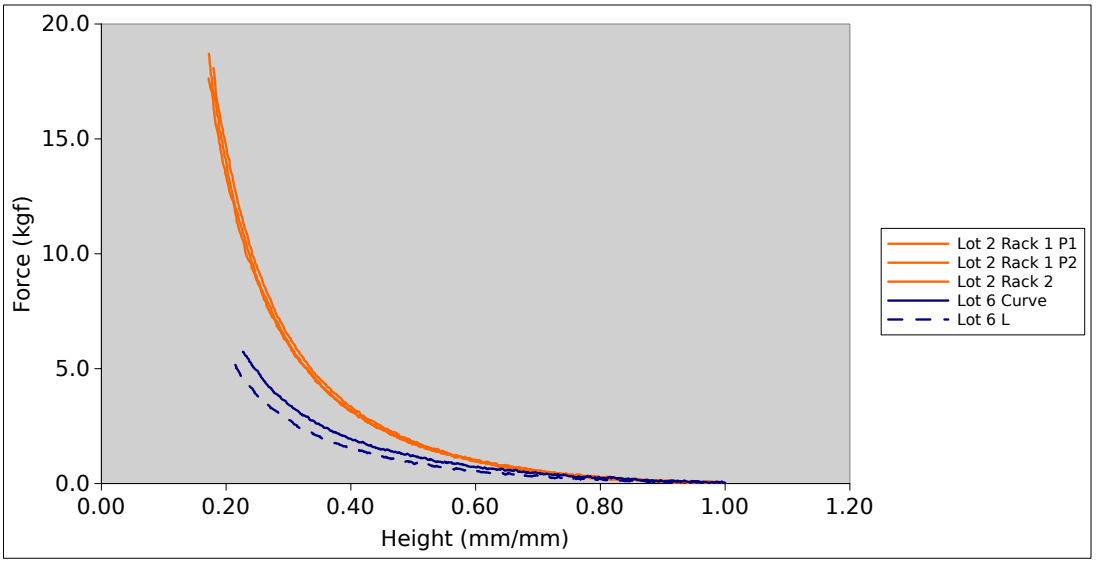


(b) Force per thickness

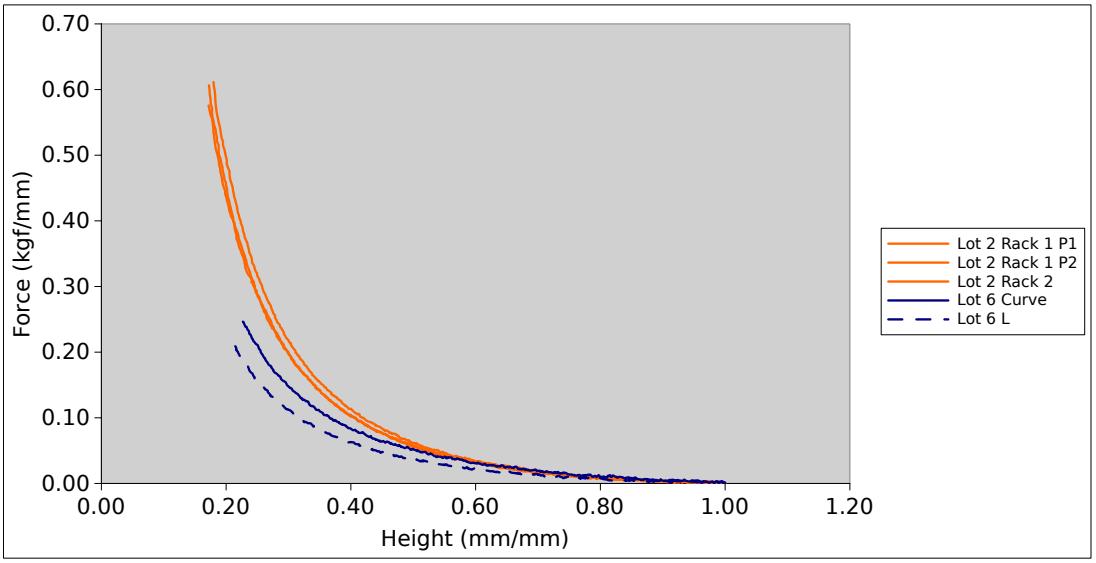


(c) Force per weight

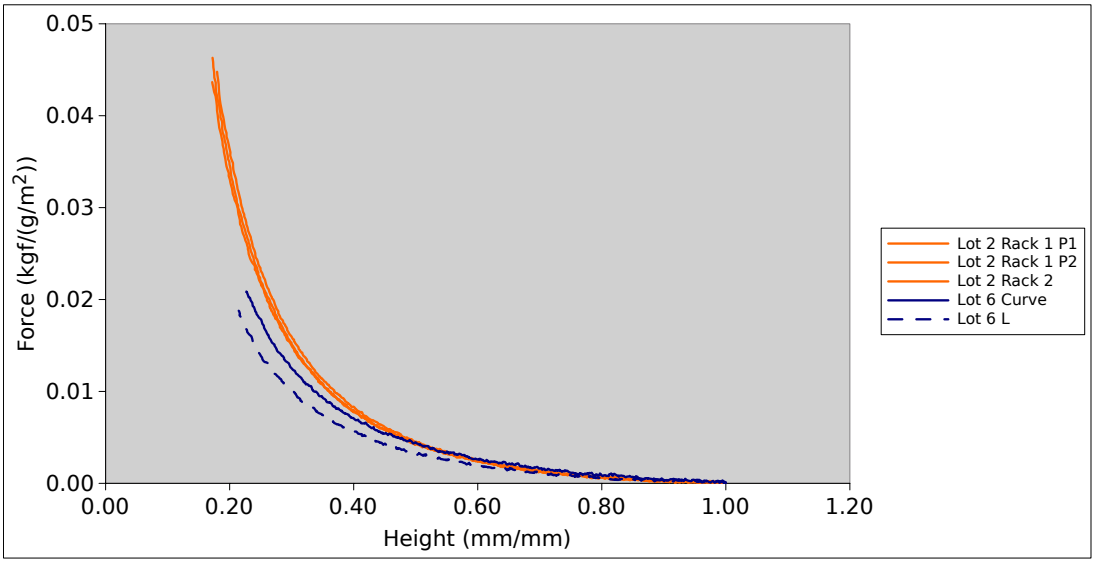
Figure 5.38: Effect on force-displacement curves of varying the wool:PLA ratio.



(a) Force



(b) Force per thickness



(c) Force per weight

Figure 5.39: Effect on force-displacement curves of additional carding.

marking the product as "dry-clean only"). However when bonded in-line with the production process, the warmth figures would be expected to rise, and thus improved washability may be feasible.

One interesting outcome of the wash tests was that hypothesis 4 has been at least partially disproved. In section 5.4.3.6 we assumed that the measured bulk loss for Lot 2 was too high; under that assumption there was no obvious correlation between PLA content and bulk loss (table 5.11b), implying that the bonded PLA substructure has no shrink-resisting effect. However, if we do not make that assumption, then the data suggests that there are two competing effects that dominate at high and low PLA content respectively.

The reduction in felting at high PLA is most likely caused by the significant reduction of wool fibre in the web, rather than any shrink-resisting effect of the bonded PLA skeleton; this again suggests that the bonded nature of the PLA has little effect.

The reduction in felting at low PLA would suggest that the number fraction of PLA also has no effect, completely invalidating hypothesis 4. However, it is possible that the PLA fibre in Lot 1 was better-blended than Lot 2; the higher fraction of finer PLA fibres may have been more difficult to card (section 2.3). Lot 2 did appear to exhibit good blending when examined via SEM (fig. 5.25), but it is difficult to draw firm conclusions from such localised observations. Even low PLA fractions by weight have significant number fractions (section 2.5.5). Thus hypothesis 4 cannot be ruled out completely given the available data, although it is by no means solid.

A major limitation of bonded carded web is their stiffness preventing drape around the body. Table 5.9a shows that knoppy web is definitely superior in this regard, and that hypothesis 2 is well-grounded. Furthermore, table 5.9b shows that the drape of the knoppy web can be easily adjusted by reducing the proportion of PLA in the web. This needs to be traded off with the ease of handling and other performance requirements. But regardless of the above deliberations about hypothesis 4, it follows from table 5.11b that reducing the proportion of PLA does not harm the washability of the knoppy web. This is a fantastic result: it indicates that, at least within the bounds of variation considered here, the drape of a knoppy web can be altered without affecting washability.

5.5 Model comparison

The previous sections of this chapter described a number of experiments, during which various force-displacement curves were measured for both knops and knoppy web. We now want to see if the various recipe parameters of the knoppy web products can be used to link our theoretical models to these experimental curves. In this section, we take the force-displacement curves from sections 5.3.1.4 and 5.4.3.7, and fit them to the models we developed in chapter 4.

5.5.1 Modelling knoppy web parameters from production variables

To fit our models to the experimental data, we need to map the model parameters to properties that were controlled or varied during the experiments. Some of the model parameters will have relationships to measurable properties that can be well-defined, while others must be determined via curve-fitting. It follows that knop properties can be determined by fitting the force-displacement curves for knops, and knoppy web parameters can be determined by fitting the force-displacement curves for knoppy web.

In section 4.2.4 we defined the following model parameters to be properties of the knop:

- r_0, the radius of the uncompressed knop.
- $2h$, the thickness of the knop wall.
- E_k, the Young's modulus of the knop membrane.
- μ_k, the Poisson's ratio of the knop membrane.

Putting aside the specifics of the wool fibre blend within the knop (which we have not investigated in this thesis), there are two key controllable properties of a knop (section 2.6):

- The size of the knop.
- The stiffness of the knop (in part tied to the ratio of PLA to wool within the knop).

It is obvious that r_0 and h should be linked to the size of the knop, while E_k and μ_k should be linked to the knop stiffness.

Likewise, in section 4.3.1.2 we defined the following model parameters to be properties of the knoppy web (beyond those parameters corresponding to the knops):

- t_0, the thickness of the web surrounding the knop.
- K, the van Wyk constant of the web.
- ρ_w, the density of the web.

In section 5.4 we measured the effect on knoppy web of the following three variables:

- The ratio of knops to web.
- The ratio of PLA to wool within the web.
- Whether the knoppy web blend was carded prior to airlay.

The effect that additional carding had on the knoppy web was to significantly reduce the size of the knops, and increase the fraction of web (section 5.4.3.1). Taking this into account, the list of relevant variables becomes:

- The size of the knops.
- The ratio of knops to web.
- The ratio of PLA to wool within the web.

The size of the knops links to the knop model parameters r_0 and h as before (although for the most part it is constant—the same knops were used for all Lots, so the only Lot with a different knop size is Lot 6). Looking at the knoppy web model parameters, t_0 can obviously be linked to the ratio of knops to web, and the stiffening effect of the PLA to wool ratio should be linked to K. ρ_w is a less obvious parameter, because increasing the web density (while holding its volume constant) will increase both the volume of the web and its stiffness. However, for simplicity, we assume that ρ_w is only linked to the ratio of knops to web.

5.5.1.1 Knop size

Mathematically, linking the size of the knops to the model is trivial—the spherical model has an outer radius $r_0 + h$. Practically however, it is much less so. Knop size is not a property that can easily be measured, for two reasons:

- There is always variation in the sizes of knops produced by the knopping process. Despite an operator's best efforts, variations in the fibre length (section 2.4) and blending (section 2.5) will lead to non-uniformity. The best that can be done is to assume an average knop size that is representative of the knops produced.

- It would be impractical to measure during production the actual sizes of knops, of a sufficient quantity to obtain an average. Doing so would also be inaccurate, because the knops would be measured under near-zero unknown loading, and the size would not reflect their actual extent within the final knoppy web.

For our purposes, we require an average measure of knop size that can be determined in a way that scales to actual production. Fortunately, the size of the knops manifests itself in their bulk, which is a commonly-measured property. We can compare the volume of a sample of knops to the total volume of the knops within it:

$$N_k \left(\frac{4}{3}\pi(r_0 + h)^3 \right) = F_k \frac{m_k}{\rho_k}, \tag{5.5}$$

where N_k is the number of knops in a sample of weight m_k, ρ_k is the average density of the knops (namely the inverse of their measured bulk), and F_k is the packing density of the knops.

For a randomly-packed arrangement of monodisperse (uniform size) spheres, $F_k \approx 0.64$ [100]. Knops are more polydispersed, and if assumed to be perfect spheres then F_k could be significantly higher. However, our knoppy web model assumes that the knops are arranged in a regular grid (assumption 4.3.1). To ensure that our model is self-consistent, we maintain that assumption here: each knop is in contact with six neighbours, and occupies a cubic volume with sides of length $2(r_0 + h)$. The packing density is therefore

$$\begin{aligned} F_k &= \frac{\frac{4}{3}\pi(r_0 + h)^3}{(2(r_0 + h))^3} \\ &= \frac{\pi}{6}. \end{aligned} \tag{5.6}$$

Maintaining assumption 4.3.1 means that the calculated knop size is likely lower than the true average, but is a better representative size for the purposes of our models. Rearranging eq. (5.5), we find that

$$r_0 + h = \left[\frac{m_k}{8 N_k \rho_k} \right]^{1/3}. \tag{5.7}$$

There is a standard test [87] that is regularly used to determine the bulk of knops, and quick enough that it is already performed during production. From there, all that is necessary is to count the number of knops N_k in some sample of weight m_k, and eq. (5.7) can then be used to determine $r_0 + h$ for use in our model.

5.5.1.2 Knop stiffness

As mentioned in section 2.6, the intrinsic stiffness of a knop (ignoring size-dependent factors) depends primarily on the composition of the fibre blend that is knopped. This can be further split into two factors:

- The choice and blend of wool fibres.
- The ratio of PLA to wool in the knop.

The issue of comparing wool blends and their effect on knops has not been studied in this thesis, and is left to future workers. We can however consider the effect of PLA in the knop, as it has been touched on incidentally in the previously-discussed work.

When a knop is present inside a bonded knoppy web, we can assume that all PLA-PLA contacts are bonded, and there are no PLA-wool or wool-wool bonds (in line with observations in section 5.4.3.3). If we wanted to mathematically link the PLA:wool ratio to model parameters, we could leverage the number ratio of the fibres (section 2.5.5), and try to calculate the number of PLA-PLA contacts formed as a percentage of the total. Each bond increases the stiffness of the PLA substructure, and thus it may be possible to determine the quantitative effect that additional contacts have on the overall knop stiffness. However, such an approach would be insufficient on its own, because it says nothing about the endpoints.

What value should the model parameters be for a knop with zero PLA, or for a pure PLA knop? In the end, these would need to be experimentally determined, and an empirical relationship derived instead of a mathematical one. Whether a general relationship (applicable beyond a specific knop) could even be determined without accounting for the wool fibre blend is unknown. It would take significant effort focused specifically on knop development to factor out the effect of changing the fibre blend on other knop properties, and so determining such a relationship is outside the scope of this thesis.

In our model, knop stiffness is represented in aggregate by E_k, the Young's modulus of the knop wall. The membrane strain term for the inner region has an additional parameter μ_k, the Poisson's ratio of the knop wall, that allows for tuning of the specific behaviour of the knop (section 4.2.5.3). These parameters essentially bundle together the effects of both the wool fibre blend and the PLA:wool ratio, and there is no clear measurable value that could be used to benchmark their values. We therefore opt to leave these as free parameters that are determined by fitting the knop model to experimental data.

5.5.1.3 Ratio of knops to web

As outlined in section 2.2, the knoppy web blend is specified as a ratio of the weight of web m_w to the weight of knops m_k. In our model, t_0 is the obvious measure of the quantity of web in the unit cell. It therefore seems logical that we could derive an equation for t_0 in terms of the weight ratio m_w/m_k.

At the outset, we run into a problem. By assumption 4.3.1 each knop effectively takes up a cube of volume, regardless of the amount of web present. $t_0 = 0$ is equivalent to having the knops packed such that they are touching one another. In this configuration, there is still free space between the knops. Equation (5.6) gave the packing density F_k of knops (or spheres in general) in a cubic lattice; the implication is that under assumption 4.3.1, over 47% of the volume of cubic-packed knops is empty space. In the case of the knoppy web unit cells given in section 4.3, there is still a significant fraction of web present at $t_0 = 0$.

The solution to this problem is to also take into account the density of the web ρ_w. In the case where $m_w/m_k = 0$, there is no web component, and by definition $\rho_w = 0$. However, there is nothing to prevent t_0 being non-zero while $m_w/m_k = 0$[3]. This leads us to the conclusion that we should instead derive an equation for ρ_w in terms of both m_w/m_k and t_0.

The mass of the knop in a knoppy web unit cell can be determined by setting $N_k = 1$ in eq. (5.7) and rearranging for m_k:

$$m_k = 8\rho_k(r_0 + h)^3. \tag{5.8}$$

The mass of web in the unit cell is the product of the web density ρ_w and the volume of web v_w, the latter of which is simply the difference between the total unit cell volume and the knop volume:

$$\begin{aligned} m_w &= \rho_w v_w \\ &= \rho_w \left(8(r_0 + h + t_0)^3 - \frac{4}{3}\pi(r_0 + h)^3 \right). \end{aligned} \tag{5.9}$$

Taking the ratio of the two and solving for ρ_w,

$$\rho_w = \frac{m_w}{m_k}\rho_k \left[\left(1 + \frac{t_0}{r_0 + h} \right)^3 - \frac{\pi}{6} \right]^{-1}. \tag{5.10}$$

It is important to recognise that ρ_w is the actual density of the web component, whereas ρ_k is the knop density averaged over the entire unit cell volume. The difference is solely algebraic under our assumptions, as ρ_k could be converted into an actual knop density if desired. The key point is that we are calculating ρ_w rather than attempting to relate it to the bulk of the web component (prior to blending with knops); between the significant differences in geometry of the web component, and the fact that the knoppy web is bonded, we believe that no practical or meaningful mapping could be made.

[3] Assuming the non-existence of gravity. But then, most physicists enjoy working in a frictionless vacuum [101], and outer space provides a convenient approximation for all three of these conditions.

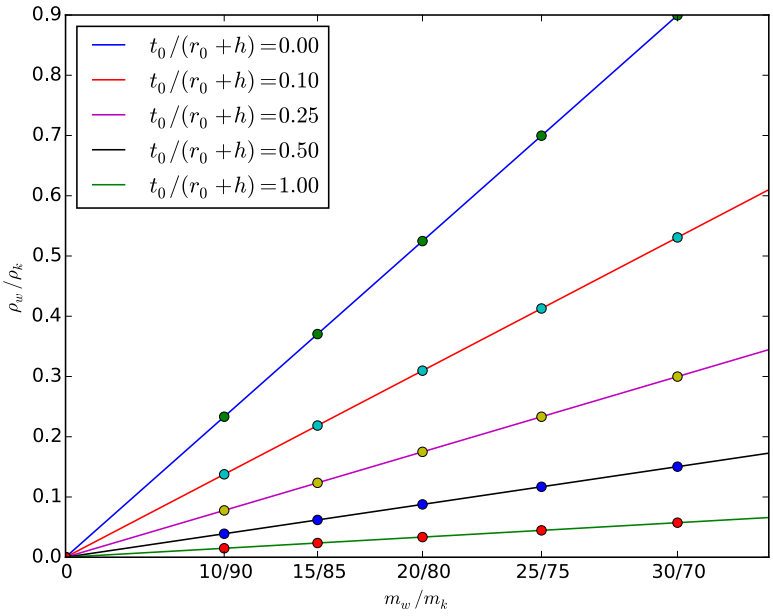


Figure 5.40: The effect of web:knop ratio and t_0 on ρ_w.

Figure 5.40 shows how ρ_w is affected by both the knop:web ratio, and the relative magnitudes of t_0 and r_0+h. ρ_k is a property of the knops, and can be considered fixed across knoppy web blends. t_0 on the other hand cannot be directly measured, at least not without some radiographic 3D imaging technique; thus we make it a fitted parameter in the model. The $t_0 = 0$ line places an upper bound on the value of ρ_w; as t_0 increases, ρ_w/ρ_k decreases. The inverse cubic relationship between ρ_w and t_0 can also be clearly observed.

5.5.1.4 Ratio of wool to PLA

Given that the web component is actually treated as a fibrous assembly, one might think that incorporating PLA-PLA bonds into the knoppy web model would be relatively easy. But none of the models considered in section 3.3 have any concept of distinct types of contact points. In fact, one of the assumptions of the van Wyk model (and many based on it) is that all contact points are bonded or fixed (by the fact that the element length does not change under compression). The closest we come to a potential solution is Carnaby and Pan's work [14] to include the effect of slippage (covered in section 3.3.2.3), but even this only has one type of contact—a contact that will slip once friction is exceeded. Extending this model with a second permanently-fixed contact point type would be useful future work.

As a first-order approximation however, we can use the relative stiffness of the knop and web components to represent the PLA content of the knoppy web. The stiffness of the web component is defined by the fibre modulus E_f and the van Wyk constant K (eq. (3.36)). E_f can be considered a property of the particular fibre blend, but is not a value that we have the capacity to measure. Therefore for the purposes of this model we set E_f to the typical fibre modulus for New Zealand wools (3.1 N/tex [72]), and use K to represent the aggregate stiffness of the web component (comprising both the PLA:wool ratio of the web and the difference in E_f of the PLA). K/E_k therefore gives the relative stiffness of the web to the knops, which can be linked to the PLA:wool ratio; holding all else constant, increasing the PLA content of the web will increase K/E_k. Formulating a general relationship is still not feasible, for the same reasons as for the knop (as outlined above). Thus we still opt to determine K by fitting to experimental data. But in this case, we do have a fixed data point against which we can make a rough comparison: the PLA content of the knop.

5.5.2 Determining knop sizes

In section 5.5.1.1 we proposed that a representative knop size (for use in our models) could be determined by counting the number of knops N_k in a sample of weight m_k. The counting process was conducted for the various types of knops used in this chapter. The following methodology was used:

1. The standard bulk of the knops was measured to determine their specific density ρ_k.
2. A 1 g sample of the bulk-tested knops was weighed out.
3. The number of knops, N_k, in the sample were spread out on a surface (fig. 5.41) and counted.
4. $r_0 + h$ was calculated from eq. (5.7).

We chose $m_k = 1$ g to strike a balance between being a large enough sample size to be representative, while small enough to be easily countable by a production line operator.

Figure 5.42 shows an example of a cluster of knops during the counting process. While there are only four fibrous groups, there are in fact six knops—two pairs of knops are joined together by fibrous tails. The test for whether a knop was distinct or not was to squash it; distinct lumps were counted as distinct knops. This visibly excludes some loose fibre, but we are assuming that the majority of the weight of fibre is contained within the counted knops.

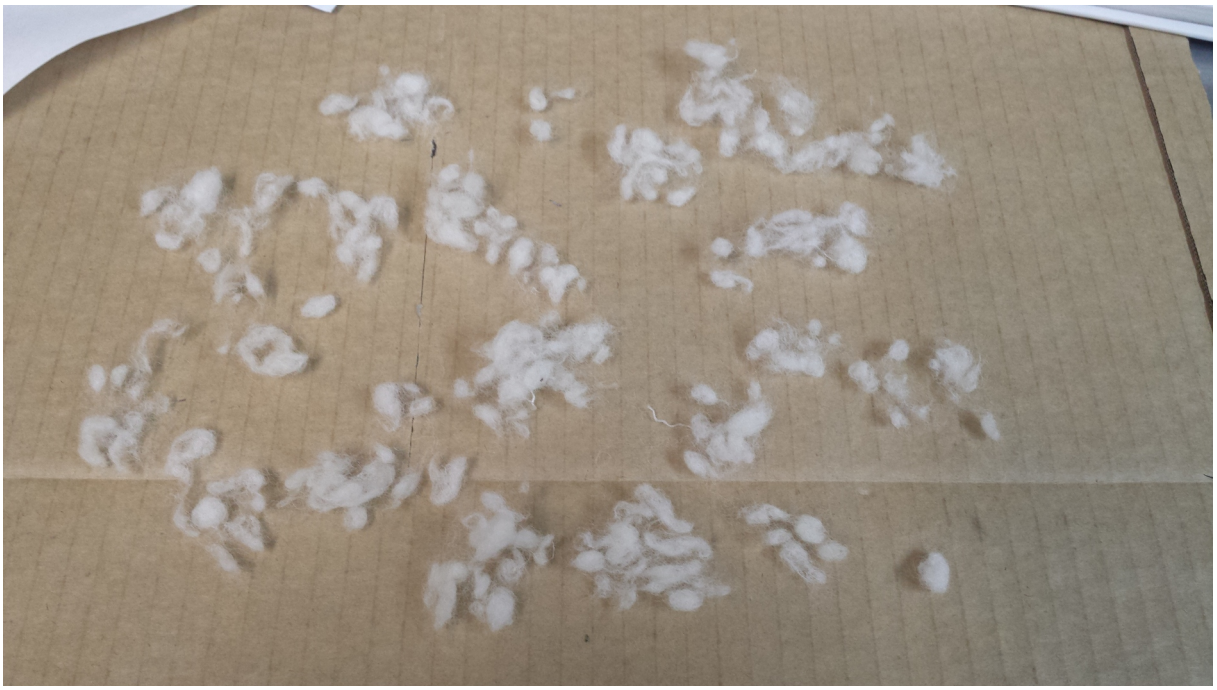


Figure 5.41: A 1 g sample of knops spread out on a surface for counting.

Figure 5.42: A group of six knops.

Table 5.12: Calculated knop diameters

Knop type	Bulk ($cm^3\,g^{-1}$)	N_k	$r_0 + h$ (cm)	Knop diameter (cm)
Long-term unbonded (section 5.3.1)	104	232	0.383	0.766
Lots (section 5.4.1)	79	202	0.366	0.732

Table 5.13: Initial values and ranges of the fitted parameters for the knop model

Parameter	Initial value	Range
h	0.005	$0.001 \leq h \leq (r_0 + h - c_{max})/2$
E_k	100	$0 \leq E_k$
μ	0.4	$0 \leq \mu \leq 0.5$

5.5.2.1 Results

Table 5.12 shows the measured knop sizes for the various types of knops used in this chapter (on which force-displacement curves were measured). The sizes all appear consistent with the observed knops, and are therefore suitable as input parameters to the models.

5.5.3 Data preparation

To obtain energy-displacement curves that could be fitted to our models, the various force-displacement curves were each integrated using the trapezoidal rule. The curves were then normalised: the displacement was divided by the initial height of the sample, and the energy was divided by the initial volume of the sample under the compression foot.

As stated in section 5.3.1.4, some knop samples had to be lightly pre-compressed to fit into the compression chamber. The energy contribution of this pre-compression is ignored.

5.5.4 Knop model fitting

The knop model fitting code was based on the knop model code described in section 4.2.5, with one key substitution: $r' = c$. The substitution arises from section 4.2, and essentially restricts the knop from expanding outwards at all. This is necessary because the knop compression data is measured in a fixed-wall chamber, preventing any lateral expansion. It also greatly shortens the runtime of the fitting code.

Table 5.13 shows the initial values of the fitted parameters, as well as the ranges used to limit them. h was given a lower limit of 0.001 cm = 10 µm, corresponding to the physically-motivated restriction that the knop wall should be at least as thick as a single fibre ($\approx$ 20 µm). The upper limit on h was also a physical limit, namely that the geometry of the model results in the requirement $c \leq r_0 - h$. E_k and μ_k were restricted to physically realistic values.

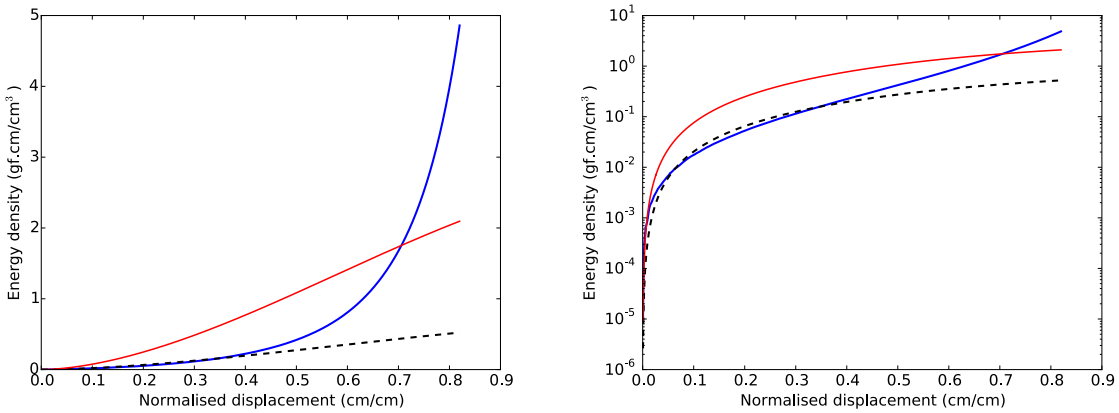
The model was fitted to the normalised data using the LMFIT curve-fitting tools [102]. Because the model parameters are non-normalised, the normalised displacement data was multiplied by the value of $r_0 + h$ for the given knop type (determined via the method outlined in section 5.5.2) to obtain the equivalent displacement of the knop. The energy calculated by the model was then divided by $8(r_0 + h)^3$ to obtain the energy density. A full code listing is provided in appendix B.7.

5.5.4.1 Results

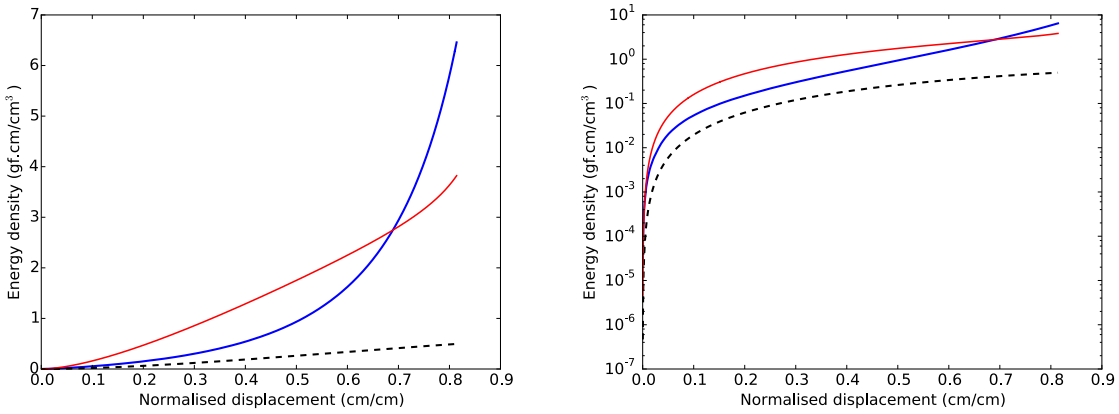
Figure 5.43 shows the results of fitting the knop model to the energy-displacement data. The knop model clearly does not provide a good overall fit for the knop compression data; the "best" fit line is nowhere near the actual curve. The χ^2 and reduced χ^2 values in table 5.14 reflect this inaccuracy. But examining the log-scale plots more closely, it is apparent that the knop model is in fact able to provide a reasonable

Table 5.14: Fitted values for the parameters of the knop model

Knop type	h	E_k	μ_k	χ^2	Reduced χ^2
Long-term bonded	0.0356	151	2.58×10^{-8}	1027	0.513
Lots	0.0022	745	0.5	809	0.403



(a) Lots



(b) Long-term bonded

Figure 5.43: Fits of the knop model to data, shown on linear (left) and log (right) plots. Blue line is data, dotted line is fit with the initial values of the parameters, red line is best fit.

fit to the data, if we restrict ourselves to low levels of compression. The knop model has a very similar curvature to the measured curves at low compression, but does not build up energy as rapidly in the later stages of compression.

5.5.5 Handling the fibrous behaviour of knops at high strain

Looking back at the original force-displacement curves in section 5.3.2.4, the high-strain behaviour appears to be quite similar to the inverse cubic behaviour of a traditional random fibre assembly. This suggests that as the cylinder of knops is compressed, the effect of the macroscopic geometry of the fibres progressively gives way to the interaction of contacting fibre segments at a micromechanical level; that is, as the density of wool fibre increases, many more contact points are formed, and the pile of knops becomes increasingly indistinguishable from a very dense fibrous batt.

To account for this within the confines of our knop model, we fill the void around each knop with a van Wyk web. As the knops are compressed, the volume of the void decreases, and van Wyk strain energy builds up, simulating the increase in fibrous strain within and around the knops. This formulation conveniently enables us to model the knop with a basic knoppy web unit cell (section 4.3.1), by setting $t_0 = 0$. The parameters of the web component (K and ρ_w) together model the strength of this fibrous effect.

The fitting code for the fibrous knop model was formed by combining the knop model code from the previous section with the van Wyk energy terms for the basic knoppy web unit cell. Table 5.15 shows the initial values of the fitted parameters for the added web component, as well as the ranges used to limit them; these were chosen somewhat arbitrarily. A full code listing is provided in appendix B.8.

5.5.5.1 Results

Figure 5.44 shows that the fibrous model provides a very close fit to the experimental data. The addition of the van Wyk component to the knop model significantly improves its accuracy. The fitted parameters (table 5.16) are also much more reasonable. The knops used in the long-term compression experiment were significantly bulkier (table 5.2) that the knops used in the ratio variation experiment (section 5.4.2.1), but both were roughly the same size (table 5.12). Thus the long-term compression knops were intrinsically stiffer, which corresponds to the larger fitted value for E_k in table 5.16.

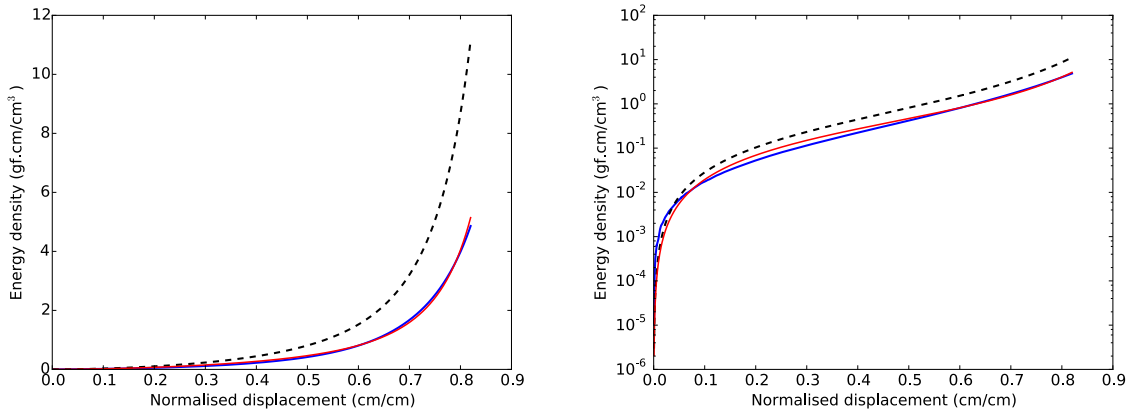
It should be noted that the fitted values for K and ρ_w in table 5.16 are minimally different, indicating that the difference in bulkiness of the knops is primarily due to the effect of the knopping. We proposed above that as a knop is compressed, its behaviour steadily becomes more like a regular fibrous assembly. Figure 5.45 shows the various energy components of the fits in fig. 5.44, and it is clear that our proposal has merit. The sphere energy term (U_S) domainates at low strain, while the energy of the added web component (U_W) dominates at high strain. Bulk measurements are conducted at relatively low pressures, and thus the observed behaviour is firmly in the U_S-dominated domain.

5.5.6 Knoppy web model fitting

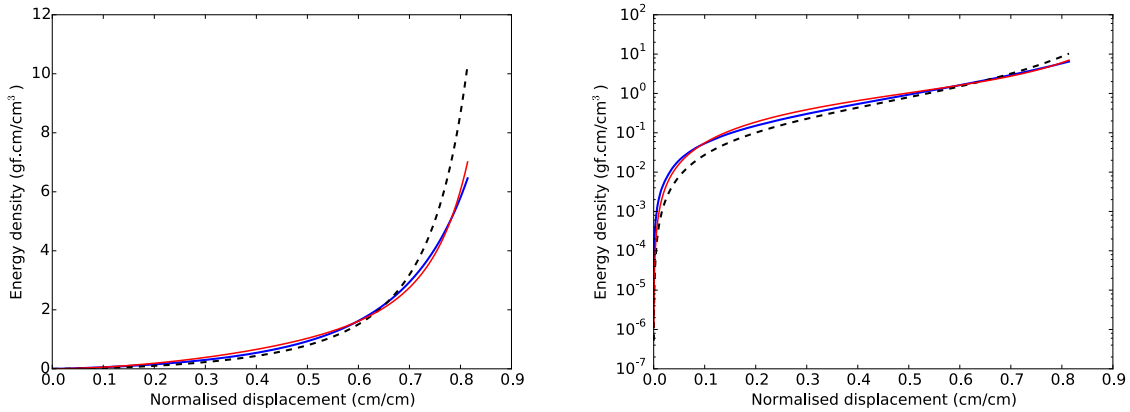
Fitting to the knoppy web curves was performed in the same manner as for the knop curves in the previous section. For each Lot, one force-displacement curve was selected from section 5.4.3.7 and fitted against the modified knoppy web unit cell. The model fitting code was originally based on the code for the modified knoppy web model, described in section 4.3.2.3. However, the performance of the code was very poor, requiring tens of hours to fit a single curve. The LMFIT curve-fitting tools run the model code hundreds of times to determine the optimum parameters. The previous brute-force method simulated the entire q_c and $q_{r'}$ parameter space to determine the minimum energy at each d for a given set of parameters. This on its own involves too many computations for the fitting code to run in a reasonable

Table 5.15: Initial values and ranges of the fitted parameters for the fibrous knop model

Parameter	Initial value	Range
K	0.001	$0 \leq K$
ρ_w	0.03	$0 \leq \rho_w$



(a) Lots

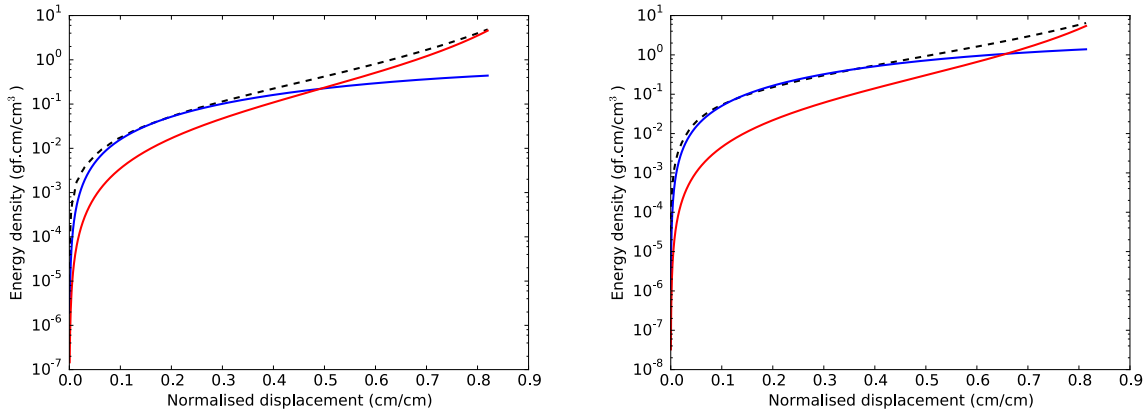


(b) Long-term bonded

Figure 5.44: Fits of the fibrous knop model to data, shown on linear (left) and log (right) plots. Blue line is data, dotted line is intial fit, red line is best fit.

Table 5.16: Fitted values for the parameters of the fibrous knop model

Knop type	h	E_k	μ_k	K	ρ_w	χ^2	Reduced χ^2
Long-term bonded	0.001	1166	0.5	7.50×10^{-4}	0.027	23.3	0.012
Lots	0.001	349	0.5	6.52×10^{-4}	0.026	5.2	0.003



(a) Lots

(b) Long-term bonded

Figure 5.45: Components of the fibrous knop model fits. Dotted line is data, blue line is U_S, red line is U_W.

timeframe, but the data curves also contain hundreds of data points. To speed up the fitting code, two modifications were made:

- The limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm variant for bound constrained optimization [103, 104] (L-BFGS-B) was used to more quickly determine the optimal q_c and $q_{r'}$ for each d. The LMFIT minimisation tools [102] were used, backed by the L-BFGS-B implementation from the SciPy [86] library.
- The data points for each curve were thinned down to about 100 points. This was a reduction of around a factor of 6. A few fitting runs were performed with more data points, and no significant difference in the fitted parameters was observed.

Table 5.17 shows the various constants used in the model fits. E_f and ρ_f are typical values for New Zealand wool fibres, as before [72]; r_0 is determined from table 5.12; the remaining values in table 5.17 are the fitted parameters from table 5.16 for the knop used.

Table 5.18 shows the initial values of the fitted parameters, as well as the ranges used to limit them. The initial values for q_c and q'_r were chosen in the middle of their range as an arbitrary starting point for the L-BFGS-B algorithm. The initial values for K and t_0 were chosen via trial and error to position the initial fits in the vicinity of the data points.

A full code listing is provided in appendix B.9.

5.5.6.1 Fitting the van Wyk model

To provide a means of comparing the modified knoppy web model to traditional models, the knoppy web compression curves were also fitted to a unit cell comprised solely of a van Wyk fibrous assembly. By choosing a unit cell with unit cross-sectional area and setting $v_0 = 1$ and $v = 1 - d$, the van Wyk energy method equation (eq. (3.36)) could be fitted directly. However, even though the code ran much faster, the van Wyk model was fitted to the same reduced dataset as the modified knoppy web model for consistency.

Table 5.19 shows the initial values of the fitted parameters, as well as the ranges used to limit them. The initial values for K and t_0 were chosen via trial and error to position the initial fits in the vicinity of the data points.

A full code listing is provided in appendix B.10.

5.5.6.2 Results

Figures 5.46 to 5.51 show the obtained fits of the modified knoppy web model and van Wyk model to the experimental data. Aside from Lot 1, the modified knoppy web model provides a reasonable fit to all of the curves. It also provides a better fit than the van Wyk model, as evidenced by the χ^2 and reduced χ^2 values in tables 5.20 and 5.21. The van Wyk model generally underestimates the stored strain energy, and in some cases fails to match the observed curvature at low strain; the modified knoppy web model, on the other hand, generally fits better at low strains (due to the knop component, per fig. 5.45). This is a pleasing result in light of the numerous simplifying assumptions we made; it suggests that we have succeeded in capturing the essence of the underlying physics.

Looking at the fitted values in table 5.20, there is a slight increase in K as the PLA content of the web is decreased (Lot 4 $\rightarrow$ Lot 2 $\rightarrow$ Lot 5). At first glance this is opposite to the trend we were expecting to see, based on our assertion in section 5.5.1 that K would be linked to the stiffness of the web component. However, this is consistent with the observations of the original force-displacement curves (fig. 5.38), namely that the PLA content in the web does not appear to have any significant bearing on the compressional strain energy that the knoppy web can store.

Table 5.17: Values of the constants for the modified knoppy web model

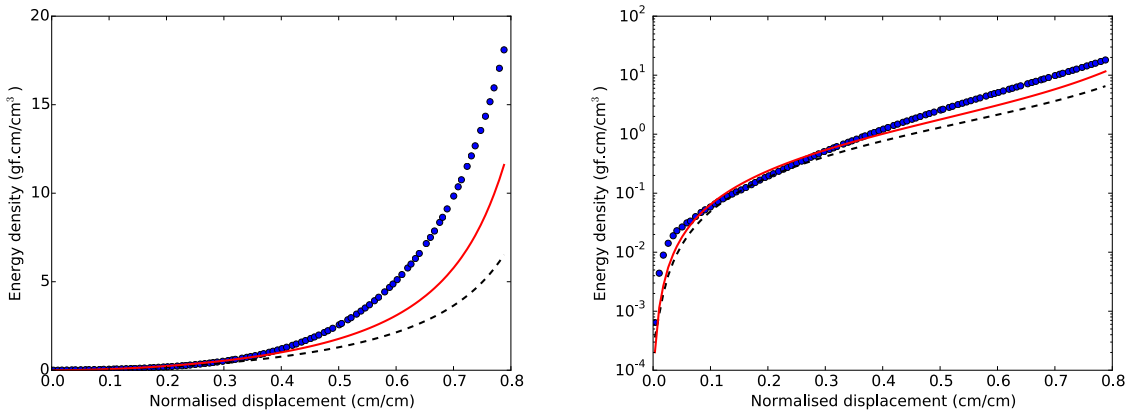
Parameter	Value	Units
ρ_k	1/79	g/cm^3
r_0	0.365	cm
h	0.001	cm
E_k	349	gf/cm^2
μ_k	0.5	
E_f	3.98×10^7	gf/cm^2
ρ_f	1.3	g/cm^3

Table 5.18: Initial values and ranges of the fitted parameters for the modified knoppy web model

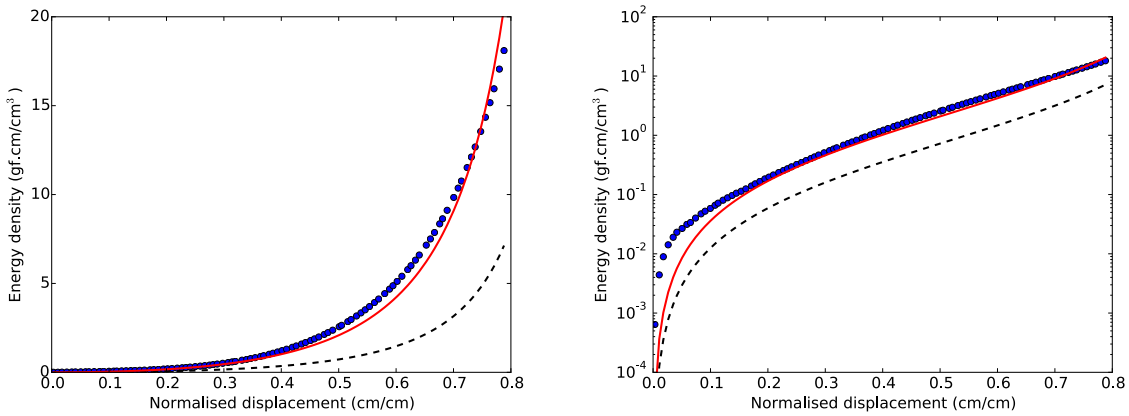
Parameter	Initial value	Range
q_c	0.5	$0.001 \leq q_c \leq 0.999$
$q_{r'}$	0.5	$0.001 \leq q_{r'} \leq 0.999$
K	10	$0 \leq K$
t_0	0.01	$0 \leq t_0$

Table 5.19: Initial values and ranges of the fitted parameters for the van Wyk model

Parameter	Initial value	Range
K	40	$0 \leq K$
ρ_w	0.001	$0 \leq \rho_w$

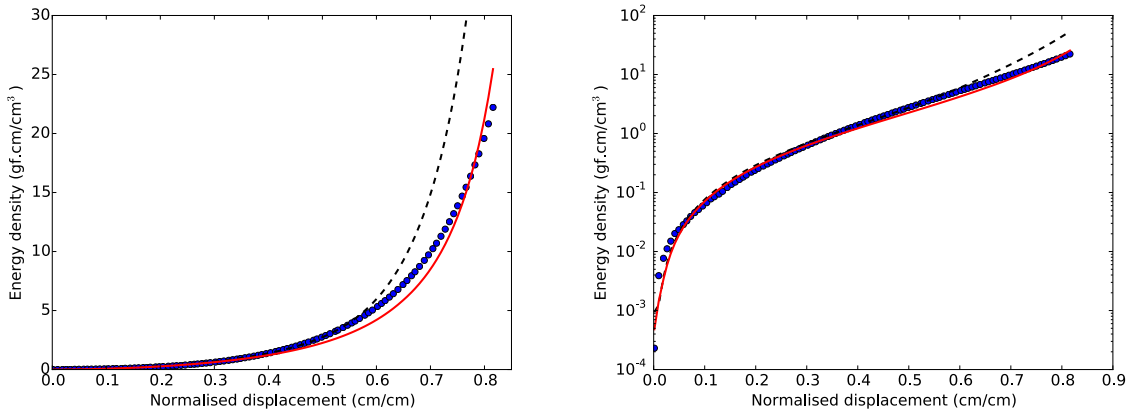


(a) Modified knoppy web model

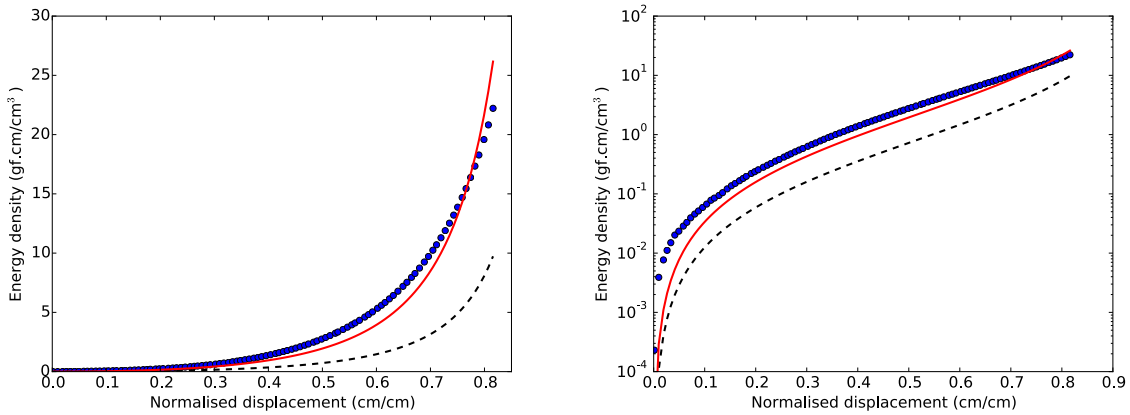


(b) Van Wyk model

Figure 5.46: Fits of the modified knoppy web model and van Wyk model to data for Lot 1, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit.



(a) Modified knoppy web model



(b) Van Wyk model

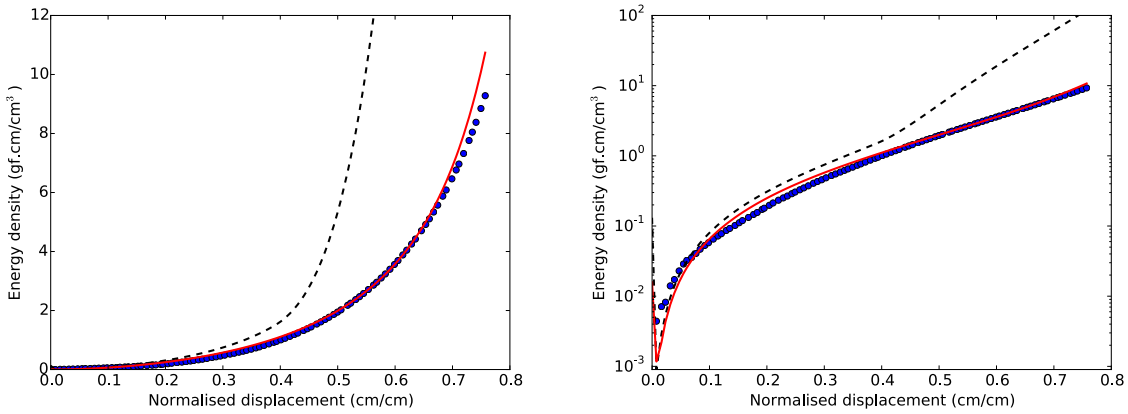
Figure 5.47: Fits of the modified knoppy web model and van Wyk model to data for Lot 2, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit.

Table 5.20: Fitted values for the parameters of the modified knoppy web model

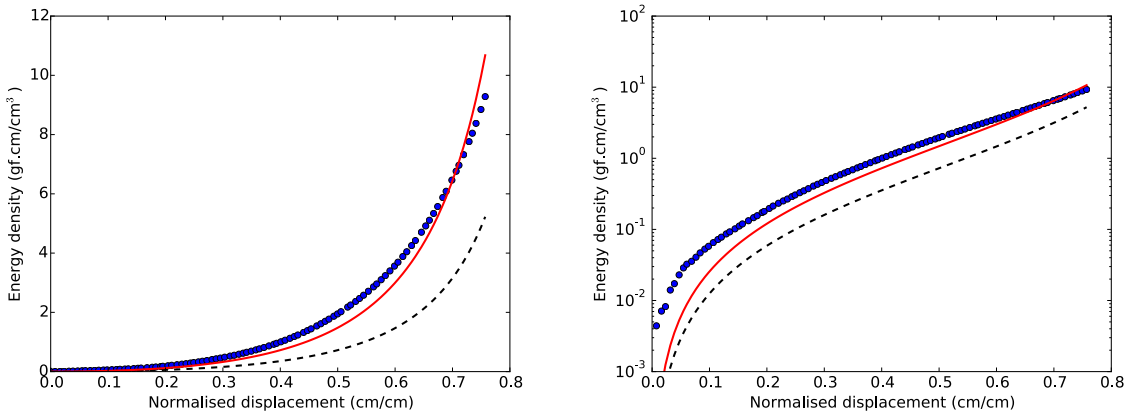
Knoppy web sample	t_0	K	χ^2	Reduced χ^2	Calculated ρ_w
Lot 1 P2	0.000491	20.699	462	4.759	0.002928
Lot 2 Rack2	0.019655	5.2436	61.6	0.604	0.004896
Lot 3 SEM P1	0.025205	0.82106	7.76	0.078	0.007777
Lot 4 #1 P1	0.022275	4.9294	88.4	0.941	0.004721
Lot 5 P1	0.024171	5.7057	29.5	0.314	0.004600
Lot 6 L	0.047424	3.6038	1.57	0.016	0.003448

Table 5.21: Fitted values for the parameters of the van Wyk energy model

Knoppy web sample	rho_w	K	χ^2	Reduced χ^2
Lot 1 P2	0.0010985	86.736	29.2	0.301
Lot 2 Rack2	0.0014078	38.549	88.6	0.868
Lot 3 SEM P1	0.0010590	68.813	13.9	0.140
Lot 4 #1 P1	0.0009826	103.99	140	1.488
Lot 5 P1	0.0010001	106.89	49.8	0.529
Lot 6 L	0.0009358	68.837	17.3	0.182

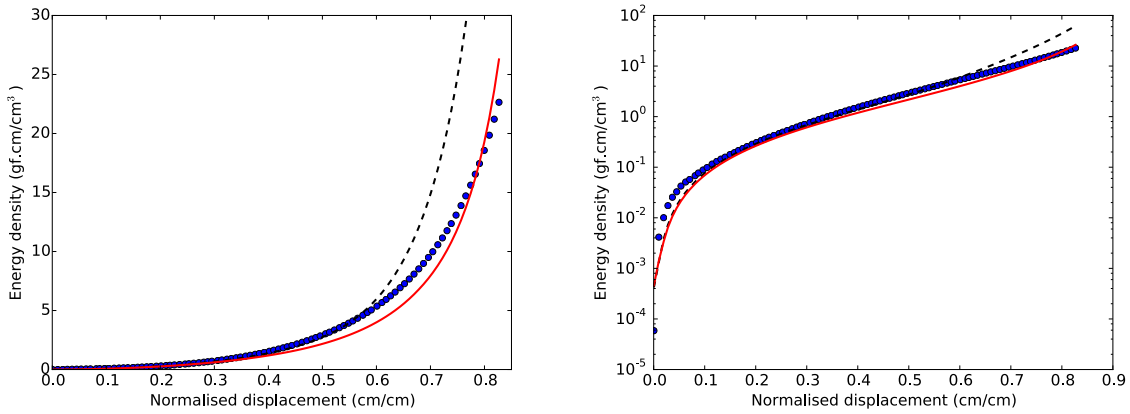


(a) Modified knoppy web model

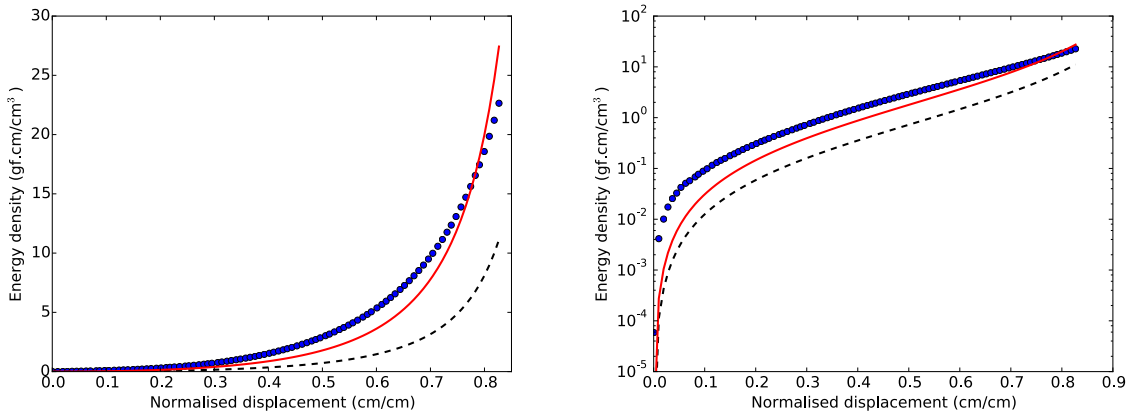


(b) Van Wyk model

Figure 5.48: Fits of the modified knoppy web model and van Wyk model to data for Lot 3, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit.

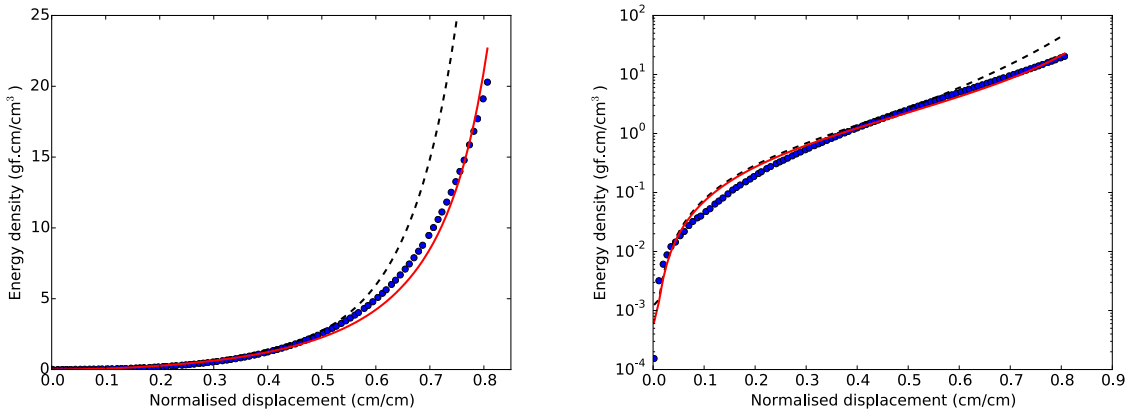


(a) Modified knoppy web model

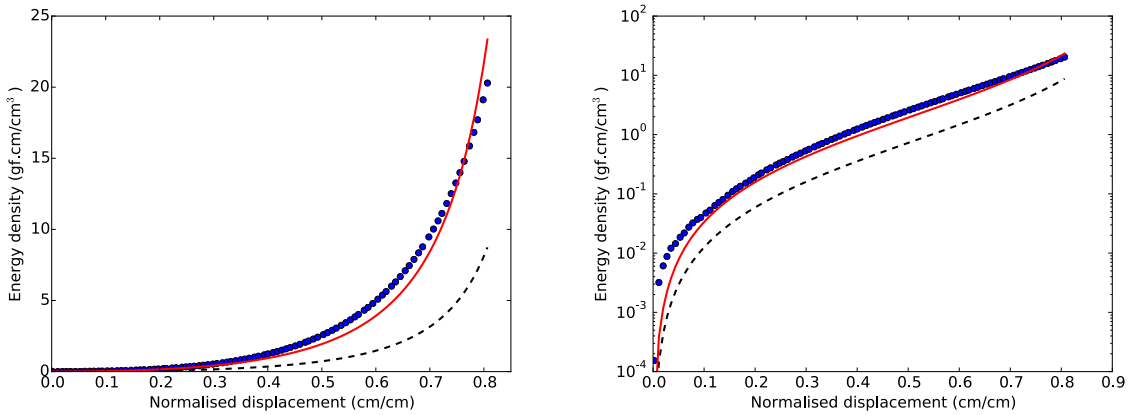


(b) Van Wyk model

Figure 5.49: Fits of the modified knoppy web model and van Wyk model to data for Lot 4, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit.

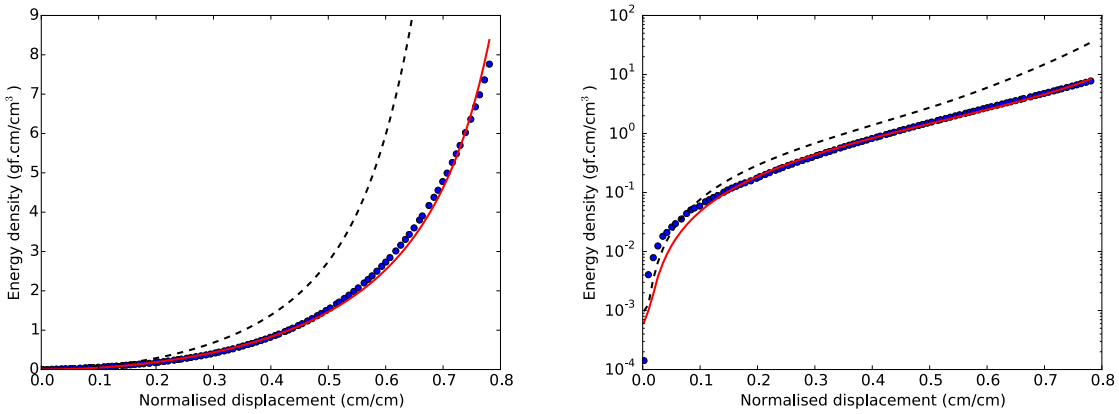


(a) Modified knoppy web model

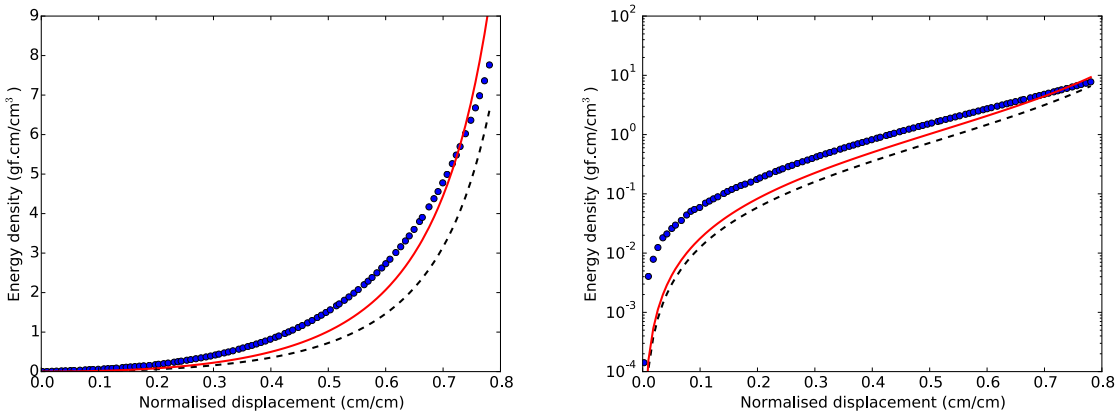


(b) Van Wyk model

Figure 5.50: Fits of the modified knoppy web model and van Wyk model to data for Lot 5, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit.



(a) Modified knoppy web model



(b) Van Wyk model

Figure 5.51: Fits of the modified knoppy web model and van Wyk model to data for Lot 6, shown on linear (left) and log (right) plots. Blue dots are the data points, dotted line is intial fit, red line is best fit.

K decreases significantly as we decrease the percentage of knops (Lot 1 $\rightarrow$ Lot 2 $\rightarrow$ Lot 3). While it is difficult to know how much of this is a real phenomenon (given the poor quality of the Lot 1 fit), we believe that it points to the primary deficiency with this model: the assumption that the knops are arranged in a cubic array (assumption 4.3.1). This is backed up by the very small values for t_0 and ρ_w. Recall that $\rho_w = 0.03$ g/cm^3 is a typical value for New Zealand wool fibres [72]; the calculated values of ρ_w are around an order of magnitude smaller. And the fitted values of t_0 essentially have the knops touching. The issue stems from the upper bound placed on ρ_w by the geometry (fig. 5.40), combined with the fact that the knoppy webs are all primarily knops. There is simply not enough web in the blends for the fitted values to be able to take on physically meaningful values. In this light, the trend in χ^2 makes sense: as the percentage of web is inceased, assumption 4.3.1 becomes more reasonable and better fits are obtained. This indicates that while the model is able to simulate the compressional behaviour of the knoppy web in most cases relatively well, its predictive power is at this stage limited.

A drop in K and χ^2 is also observed when additional carding is applied (Lot 2 $\rightarrow$ Lot 6). This fits with the above reasoning, because the carding process converts much of the knop fibre into web fibre (section 5.4.3.1), to the point where the Lot 6 fit is the most accurate. The drop in K is not as large as for Lot 3, but this can be explained by recognising that the "true" value of r_0 in Lot 6 is smaller than the simulation was run with; thus the fitted values of t_0 and K are correspondingly larger due to the fact that the model was fitted to normalised data.

5.6 Product development

One of the key issues raised in section 1.3 was that the knoppy web product needed to be optimised for different uses. In the final section of this chapter, we take the knowledge gained in the previous sections and use it to establish knoppy web specifications for use in overbody, underbody and apparel applications. We then outline the production of end products from these specifications, and present the results of initial consumer trials. The trials have shown that while there are some weaknesses that need to be resolved, knoppy web is a viable and promising new technology for producing desirable products.

5.6.1 Development of specifications

5.6.1.1 Overbody

For overbody products the key desired properties are:

- Stability (the product should not slide around over the user).
- Warmth.
- Loft.
- Softness.

Of the specifications studied in section 5.4, the best loft (bulk) was found in Lot 6, followed by Lot 3 (table 5.10); this pattern was reflected in the warmth to mass and warmth to thickness ratios (table 5.7). Lot 6 also provided the softest feel (section 5.4.3.1). However, Lot 5 had the lowest bending modulus by far (table 5.9b), and therefore the best drape.

Stability of an overbody product is an important property, and is closely linked to drape—a product that drapes over the user is less likely to slide around. The Lot 5 specification was therefore chosen as the base for the optimum overbody blend, and a light carding step (based on the Lot 6 specification) was added to provide extra opening.

Upon further reflection and discussion, various modifications to this blend were made:

- Short fine lambswool was introduced into the blend. It can be sourced within New Zealand, and acts to soften the final product.

- The "feel" of the samples containing 25% PLA in the web was deemed to be unsuitable for overbody use. As shown in table 5.9b, the ratio of wool to PLA in the web can be directly correlated to the stiffness of the knoppy web. To ensure that the resulting duvets would be of sufficient drape, the percentage of PLA in the web was reduced down to 15%.

- The percentage of knops was decreased from 80% to 75%. The motivation behind this change was to reduce the feel of the knops in the product—fewer knops with more web fibre around them will be less noticeable to end users. It also decreases production time, as there are fewer knops to produce. The increase in stiffness caused by this change (table 5.9a) is outweighed by the above decrease in web PLA content (table 5.9b). The decrease in resilience is minimal (fig. 5.37), but resilience is not a key property for overbody usage.

- Production time and cost was decreased by using the same fibre components and wool:PLA ratio in both the knops and the web. The creation of a particular fibre blend ratio is already a multi-step process, necessary to aid in the effective blending of the PLA fibre through the wool (as discussed in section 2.5). Using a different wool:PLA ratio in the knops to the web is efficient if the same knops are to be utilised in a variety of products, and thus can be produced in bulk. However, experience has shown that in many cases, the fibre specifications of the knop can be as important for user-facing properties (such as handle and feel) as the web they are contained in. Using the same base wool blend for knops and web halves the number of fibre blending steps, and means that knops are produced for-purpose. This did mean moving to the use of 51 mm bicomponent PLA in the knop, which means that the cutting process will result in a smaller fraction of fibre lengths less than 15 mm (section 2.4).

The extra opening step was also removed during the first production run. "Light carding" is difficult to achieve in practice: the workers had been raised for the production of Lot 6, and that still caused near-complete destruction of the knops (fig. 5.24) and a corresponding loss in resilience (fig. 5.39). We decided that preservation of the knops was more important than the blending and softening that the light carding would potentially bring. Additionally, the knoppy web appeared to be blended well with the web fibre after running it through the opener, removing the need for additional blending. The knops were softer due to the inclusion of the lambs wool, and we believe that they blend better at the opening step because their specific gravity more closely matches the web fibre.

5.6.1.2 Underbody

For underbody products the key desired properties are:

- Comfort.

- Resilience.

To fulfil the comfort requirement, the underbody specification was based on the overbody specification. Varying the wool:PLA content of the web has no noticeable effect on resilience (fig. 5.38). The fraction of knops in the blend could have been increased to increase resilience, but the effect of this was not considered to be sufficient (fig. 5.37) and would compromise on comfort. Thus the resilience requirement was instead fulfilled by modifying the overbody spec to be significantly denser. The wool blend was altered to use the coarser 607/8's wool alongside the high bulk Downwool, and the knopping process was adjusted to produce larger and more resilient knops.

Table 5.22: Optimum specifications for knoppy web blends.

(a) Overbody:	Wool	75% high bulk down wool (33.5 µm) 25% short fine lambswool (27 µm)
	PLA	51mm bicomponent (4 Denier)
	Wool:PLA	85:15
	Knops:Web	75:25
	Target weight	275 g m^{-2}
(b) Underbody:	Wool	50% high bulk down wool 50% 607/8's crossbred bellies and pieces (34.6 µm)
	PLA	51 mm bicomponent
	Wool:PLA	85:15
	Knops:Web	75:25
	Target weight	700 g m^{-2}
(c) Apparel:	Wool	100% short fine lambswool
	PLA	32 mm bicomponent
	Wool:PLA	85:15
	Knops:Web	75:25
	Target weight	150 g m^{-2}

5.6.1.3 Apparel

For apparel products the key desired properties are:

- The handle must be very soft.
- The knops must not be "noticeable."
- The knoppy web must be light ($\sim 150\,\mathrm{g\,m^{-2}}$).

The apparel specification was based on the overbody specification, for the same reason as the underbody—the overbody specification ratios provide an effective balance of comfort and performance. To achieve the additional softness and lightness required, the apparel blend was moved in the opposite direction to the underbody blend, making it less dense. The high-bulk down wool was dropped from the specification, making the 27 µm lambswool the primary base for the web component.

The requirement on "noticeability of the knops" is stronger than for overbody products, because apparel is in close contact with users for extended periods of time while they are awake. The size of the knops is a big factor in whether they are noticeable. As mentioned in section 2.6, one of the primary factors in the size of the knops is the length of the fibres. The 27 µm lambswool is ideal because its inherent length distribution is short (2.5 cm to 3 cm); nevertheless the cutting length was made as short as mechanically possible using the available machinery (about 25 mm). The 51 mm PLA fibre was also replaced with 32 mm fibre to better match the length distribution of the lambswool.

Carbonized noils were proposed as an additional component of the wool blend; the fibre is very short (1 cm to 2 cm) and fine (approx. 20 µm), meaning that in addition to producing small knops, the knops would be softer (because for a fibre of diameter D, bending stiffness is proportional to D^4 [105]). However, noils are a non-standard product that cannot be sourced within New Zealand. Although the base cost of noils is relatively cheap, the cost and hassle of importing combined with their inherent variability makes designing a reproducible knoppy web blend intractable. They were therefore left out of the final specification.

5.6.2 Optimum knoppy web specifications

As a result of the discussions in the previous section, "optimum" specifications for the overbody, underbody and apparel knoppy webs have been created. These specifications are presented in table 5.22. The target

weights were chosen based on industry knowledge of the product markets that were being targeted. The production processes for these specifications are based on the standard production process outlined in fig. 2.1.

Figure 5.52 shows a flow diagram describing the production process for the overbody knoppy web specified in table 5.22(a). The short fine lambswool was blended with the PLA, partly because the fibres were both fine, and partly because the required weights were similar. This blend was then cut to shift the overall fibre length distribution towards smaller lengths. The high bulk Downwool was separately blended and cut; then the two were blended together.

Figure 5.53 shows a flow diagram describing the production process for the underbody knoppy web specified in table 5.22(b). The process was generally similar to the overbody process. The primary difference was in the inital blending; because a 50:50 ratio of high bulk Downwool to 607/8's was being used, this was pre-opened together and then split for blending with PLA.

Figure 5.54 shows a flow diagram describing the production process for the apparel knoppy web specified in table 5.22(c). With all of the fibres in the specification being very fine, blending is an important step, and therefore considerable pre-melling (section 2.5.3) was performed. The wool component was opened on its own, and then split to be blended with the PLA. Instead of blending equal weights, the pre-melling ratio for the initial wool-PLA blend was chosen to enable the carded slivers to be passed through the cutter together in a 1:1 ratio (section 2.5.4), improving the blend.

5.6.3 Methods

5.6.3.1 Knoppy web production

The optimum specifications given in the previous section were each used to produce rolls of knoppy web. The rolls were bonded offsite at 140 °C for 5 min in a continuous bonding oven.

Two weighed samples of the overbody knoppy web gave an average weight of $342\,\mathrm{g\,m^{-2}}$, while two weighed samples of the underbody knoppy web gave an average weight of $598\,\mathrm{g\,m^{-2}}$. These differ from the target weights specified in section 5.6.2, but as explained in section 2.7.1 it is difficult to meet target weights exactly.

5.6.3.2 Sample products

The rolls of bonded underbody knoppy web were used to produce nine baby mattresses. The mattresses were the same shape and size as FibreTech New Zealand Ltd's regular knop-filled baby mattresses (fig. 5.55), and each contained three layers of underbody knoppy web (fig. 5.56).

The roll of bonded overbody knoppy web was used to produce twelve singles duvets. A single layer of knoppy web was used in the duvets, and one of three different fabrics (all lightweight cotton woven fabric) was used on the outside of the duvets. The duvets were stitched on a Dolphin digital quilter. Three of the duvets used a novel stitch pattern that was developed during a concurrent design-oriented study [106]. The remaining duvets were stitched using the standard grid pattern (fig. 5.57).

The apparel knoppy web was sent to a clothing manufacturer to produce a sample quilted jacket, shown in fig. 5.58. As a sample, the jacket was very promising; it had a luxurious soft handle, with no obvious lumpiness due to the knops.

5.6.3.3 Stiffness

Samples of underbody and overbody knoppy web (from the product run used in the consumer trials) were tested to determine their bending stiffness, using the methodology specified in section 5.4.2.5. Also tested were two sheets of apparel knoppy web provided by the factory (from the same batch, but otherwise of uncertain provenance), and a sample of the downproof fabric that is often used as a covering when quilting the knoppy web.

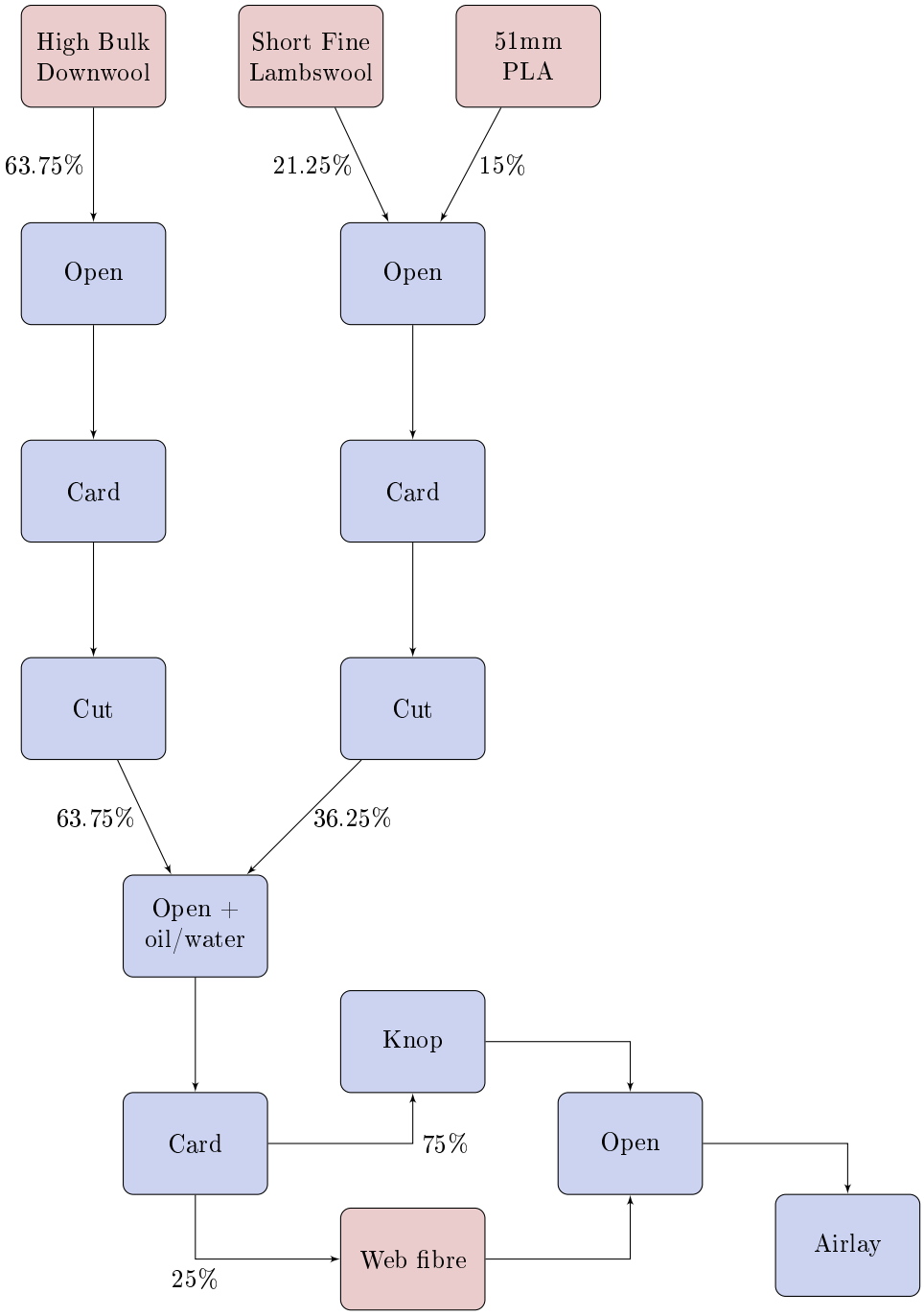


Figure 5.52: Production process for overbody knoppy web.

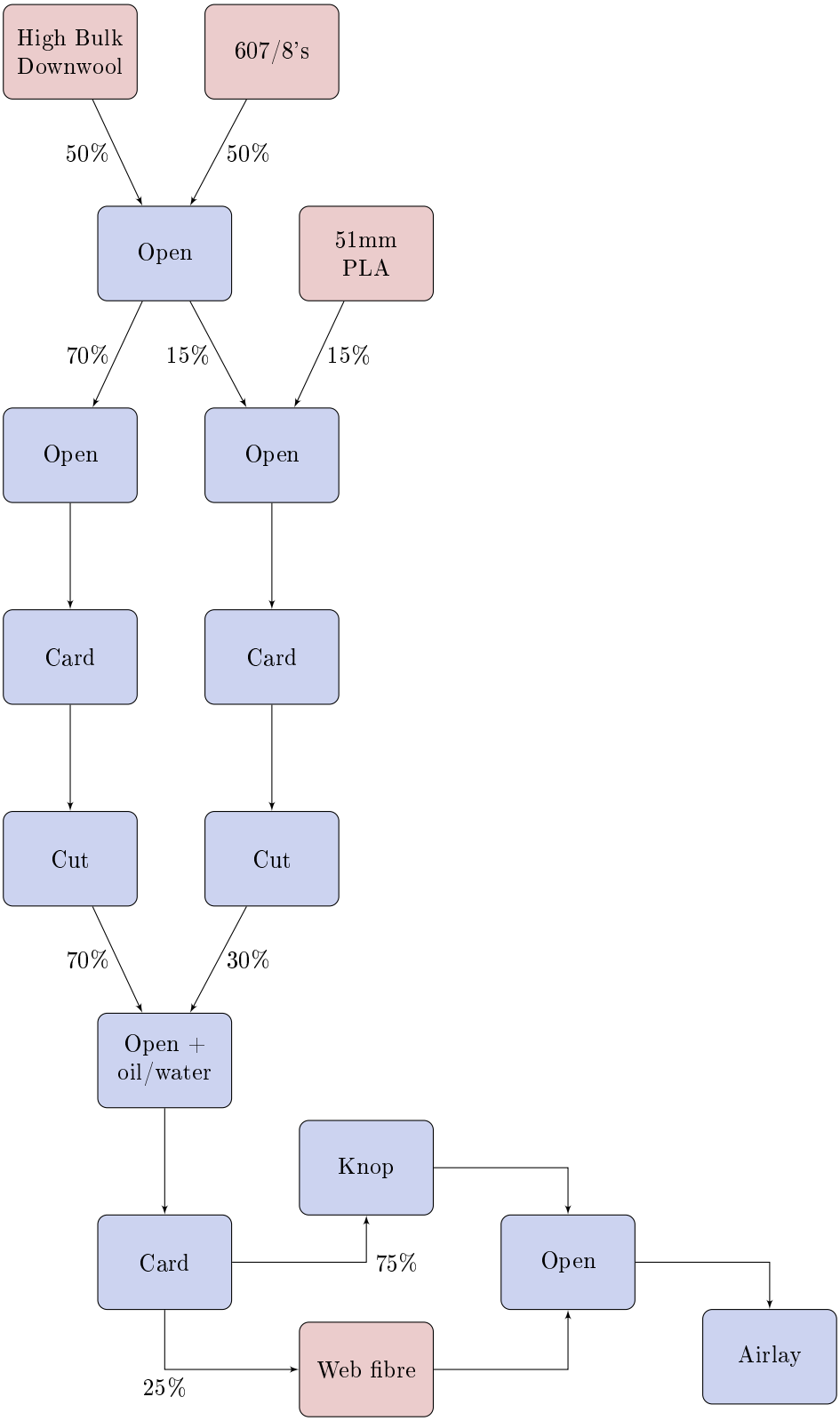


Figure 5.53: Production process for underbody knoppy web.

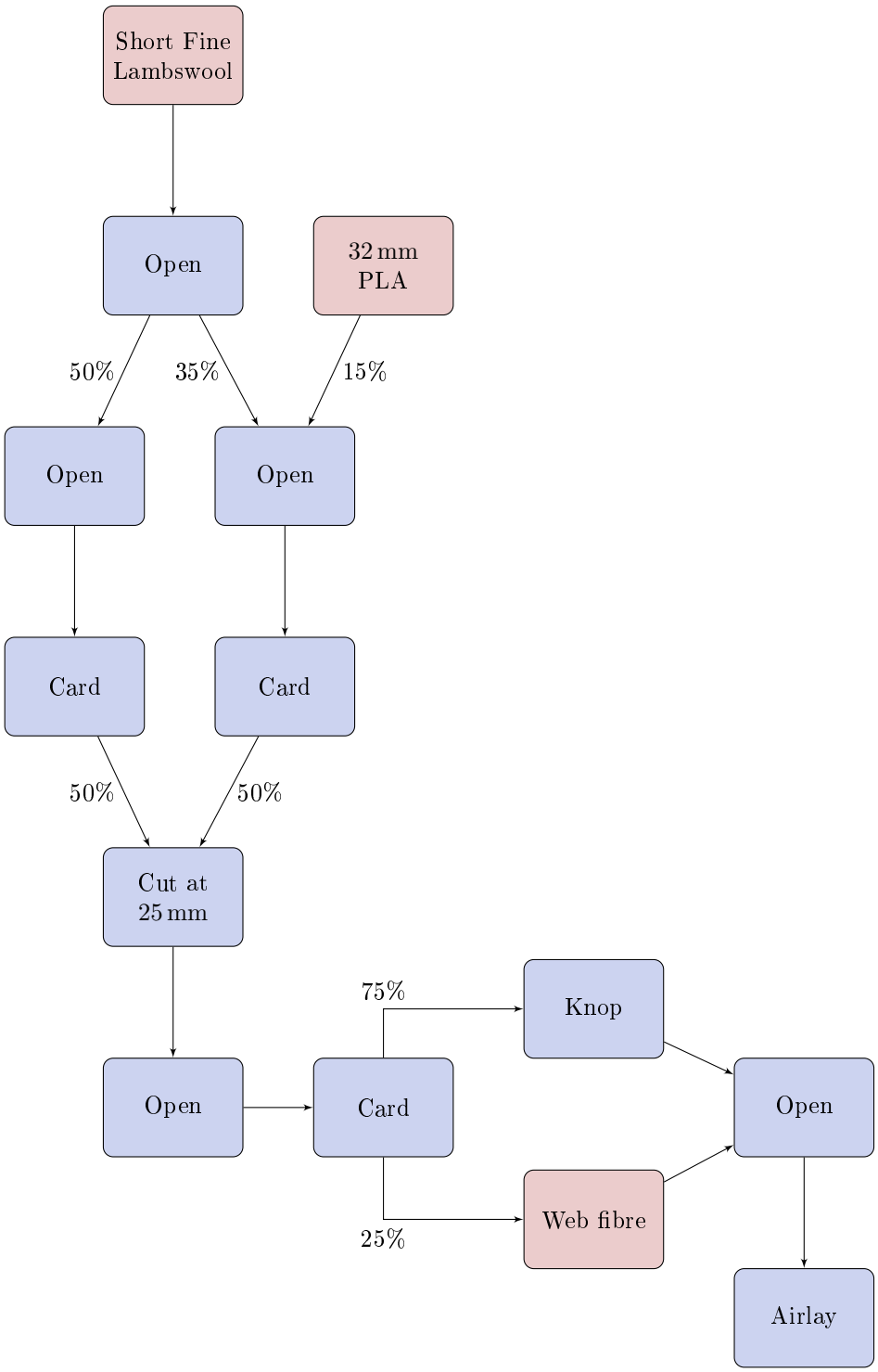


Figure 5.54: Production process design for apparel knoppy web.

Figure 5.55: One of the baby mattresses produced from the underbody knoppy web.

Figure 5.56: The baby mattress contains three layers of underbody knoppy web sandwiched within a standard shell.

5.6.3.4 Force-displacement curves

Force-displacement curves were measured for unbonded samples of the overbody and underbody knops, using the methodology described in section 5.3.1.4. Bonded knops were not used because the sample size was small, and while fig. 5.17 did show that bonded knops have better performance at large strain than unbonded knops, for a single compression cycle the bonded and unbonded knops both recover to the same level. Thus a comparison of the compression curves for unbonded knops will give a reasonable indication of their relative behaviour.

Force-displacement curves were measured for samples of the overbody and underbody knoppy web, using the methodology described in section 5.4.2.7. Additionally, two "test" samples were also measured; these samples were taken during the airlay weight determination process (section 2.7.1) for the overbody knoppy web, and therefore are identical in composition but differ in weight.

5.6.3.5 Consumer trial of duvets

A consumer trial of the duvets was conducted by Rebekah Harman at FibreTech New Zealand Ltd [107]. This was the first time that knoppy web had been manufactured into a duvet; thus the focus of the trial was to gain initial user feedback, and in particular to identify any issues with the overbody knoppy web before proceeding to market.

The trial was conducted over a five-month period through spring and summer. Participants in the trial were given a questionnaire about their existing bedding. They were then given the knoppy web duvet to trial for a three month period, after which the same questionnaire was repeated. The questionnaire asked about six key performance areas:

- Drape over body.
- Weight.
- Warmth.
- Touch and feel.
- Bedding stability.
- Overall comfort.

Participants were asked to rate each of these areas on a scale of 1 to 7 (with 1 being very poor and 7 being excellent), and to provide verbal feedback.

5.6.4 Results

5.6.4.1 Stiffness

Table 5.23 gives the measured properties of the tested samples, and the calculated values for the bending length c, flexural rigidity G and bending modulus q for each sample. Unfortunately, the batch specifications of the apparel knoppy web were not known, and so the weight of the samples had to be estimated. Approximate calculations on the (non-square) sheets gave a weight of $146\,\mathrm{g\,m^{-2}}$, however measurements made of the actual bending strips tested gave weights of $127\,\mathrm{g\,m^{-2}}$ and $129\,\mathrm{g\,m^{-2}}$. Calculations have been done using both estimated weights.

The first thing to note is that the different weight measurements for the apparel samples have a small but noticeable effect on the calculated values. Looking at q in particular, the change due to a nearly $20\,\mathrm{g\,m^{-2}}$ difference in weight is roughly the same as the change due to a 0.1 cm change in sample thickness. This implies that the thickness of the knoppy web has a greater effect on drape than the weight. However, it does not mean that the bending stiffness is necessarily resistant to variations in

Figure 5.57: One of the single duvets produced from the overbody knoppy web. The duvet has been cut open to show the layers of knoppy web and fabric.

Figure 5.58: One of the quilted jackets produced from the apparel knoppy web.

Table 5.23: Stiffness properties of the various Lots across the airlay

Sample	Weight ($\mathrm{g\,m^{-2}}$)	Est. thickness (cm)	Avg. length (cm)
Underbody	598 ± 8	6.0 ± 0.25	17.3 ± 0.1
Overbody	342 ± 4	3.0 ± 0.25	10.1 ± 0.1
Apparel 1	129 ± 6 146 ± 12	1.4 ± 0.2	7.1 ± 0.1
Apparel 2	127 ± 6 146 ± 12	1.3 ± 0.2	7.1 ± 0.1
Downproof	137 ± 4	0.025 ± 0.006	4.31 ± 0.02

(a) Measured properties

c (cm)	G ($\mathrm{g\,cm}$)	q ($\mathrm{g\,cm^{-2}}$)
8.65 ± 0.05	38.7 ± 1.2	2.2 ± 0.3
5.05 ± 0.05	4.4 ± 0.2	2.0 ± 0.6
3.55 ± 0.05	0.6 ± 0.1 0.7 ± 0.1	2.5 ± 1.3 2.9 ± 1.6
3.55 ± 0.05	0.6 ± 0.1 0.7 ± 0.1	3.1 ± 1.7 3.6 ± 2.1
2.15 ± 0.01	0.137 ± 0.006	$(11 \pm 8) \times 10^4$

(b) Stiffness parameters

Table 5.24: Weights and thicknesses used in calculation of figs. 5.60b and 5.60c.

Sample	Weight ($g\,m^{-2}$)	P1 thickness (mm)	P2 thickness (mm)
Test 1	453	37.24	38.34
Test 2	250	20.77	—
Overbody	342	30.43	31.69
Underbody	598	53.34	55.90

weight, as the calculation is inherently assuming an unchanged bending length. In reality a knoppy web with higher weight but identical thickness would be denser and therefore bend less easily.

Looking at table 5.23b, the overbody sample has a smaller q than the apparel samples. But per the discussion in section 5.4.2.5, this only means that if the apparel samples were the same thickness as the overbody sample (say, by doubling up the apparel knoppy web prior to bonding), they would indeed be stiffer. The reality is clearly apparent in the flexural rigidities—the G value for the overbody sample is an order of magnitude larger than the apparel samples.

Comparing table 5.23b to table 5.8b, there is clear evidence that the apparel samples have very good drape: the flexural rigidity of the apparel samples is between one and two orders of magnitude lower than for all of the Lots. Looking at the bending modulus, there is minimal difference in intrinsic stiffness between the underbody, overbody and apparel samples; recalling the observations in tables 5.9a and 5.9b, this is consistent with the fact that all three samples have the same knop:web and wool:PLA ratios (table 5.22).

The downproof sample provides an extreme example of how G and q differ. The downproof fabric has the lowest flexural rigidity of all the samples tested, which agrees with qualitative assessments. But it has a bending modulus that is several orders of magnitude larger than any of the knoppy web samples. Thus the downproof fabric is intrinsically much stiffer than knoppy web, but wins on drape by sheer virtue of being significantly thinner.

This has implications for the use of downproof fabric as a cover for quilting. The fabric on the underside of a quilted knoppy web will have no effect on the drape, as the fabric is directly against the bending edge. However, the fabric on the topside of the knoppy web is raised above the bending edge (by the thickness of the knoppy web). This magnifies the effect of the intrinsic stiffness of the downproof fabric. The upshot is that the drape of quilted knoppy web can be noticeably compromised by enclosing it in a fabric.

5.6.4.2 Force-displacement curves

Figure 5.59 shows the force-displacement curves for the overbody and underbody knops. What is striking about this figure is that the underbody knops were produced from a blend designed to be bulkier than the overbody blend (section 5.6.1.2), yet they are in fact slightly less resilient to compression than the overbody knops, and exhibit less recovery.

Figure 5.60 shows the force-displacement curves for the various samples. The force-per-thickness and force-per-weight curves were calculated using the properties in table 5.24. The thicknesses were taken as the height of the Instron foot when the load cell first registered a positive force. The weights for the overbody and underbody samples are from section 5.6.3.1. The Test 1 sample had a measured weight of $453\,g\,m^{-2}$. The Test 2 sample was measured at $220\,g\,m^{-2}$; however after the airlay had been adjusted and the overbody knoppy web produced, an 8 g error was discovered on the scales, and the Test 2 sample weight was recalculated at $250\,g\,m^{-2}$.

As was intimated in section 2.7.1, it is clear that varying the airlay weight does not have a reliable effect on the compression properties of the resulting knoppy web—otherwise, the curves for the overbody sample and test samples in fig. 5.60c would exactly overlay each other. The change in airlay settings between Test 1 and Test 2 has resulted in a thinner, lower-weight sample that appears to perform identically (fig. 5.60a),

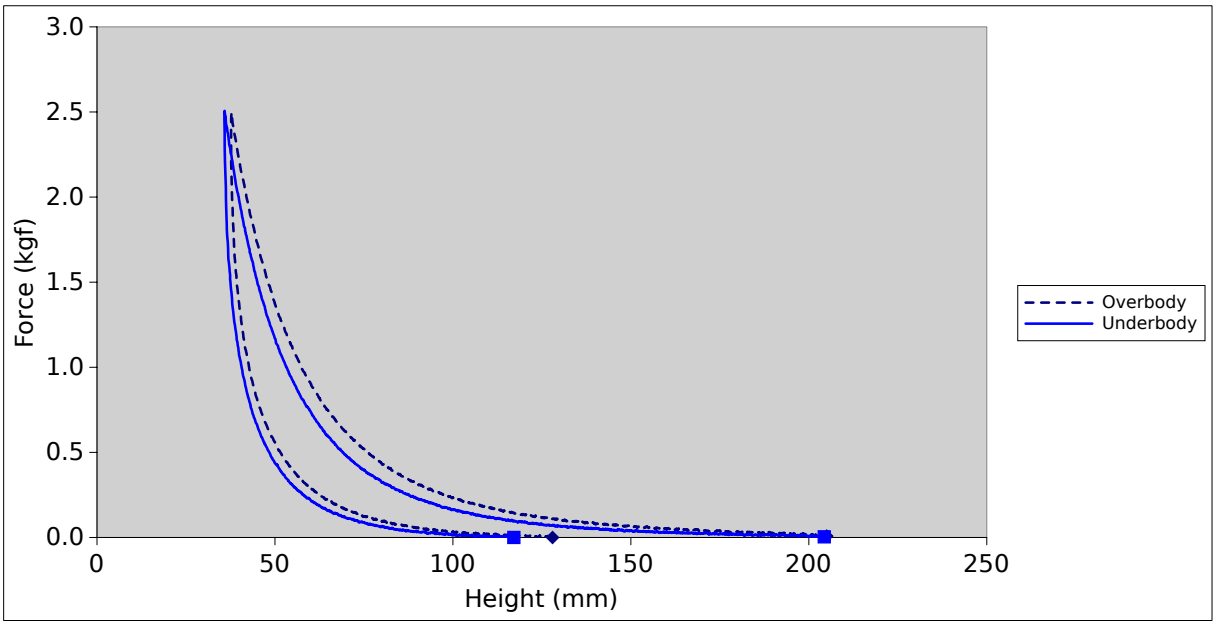
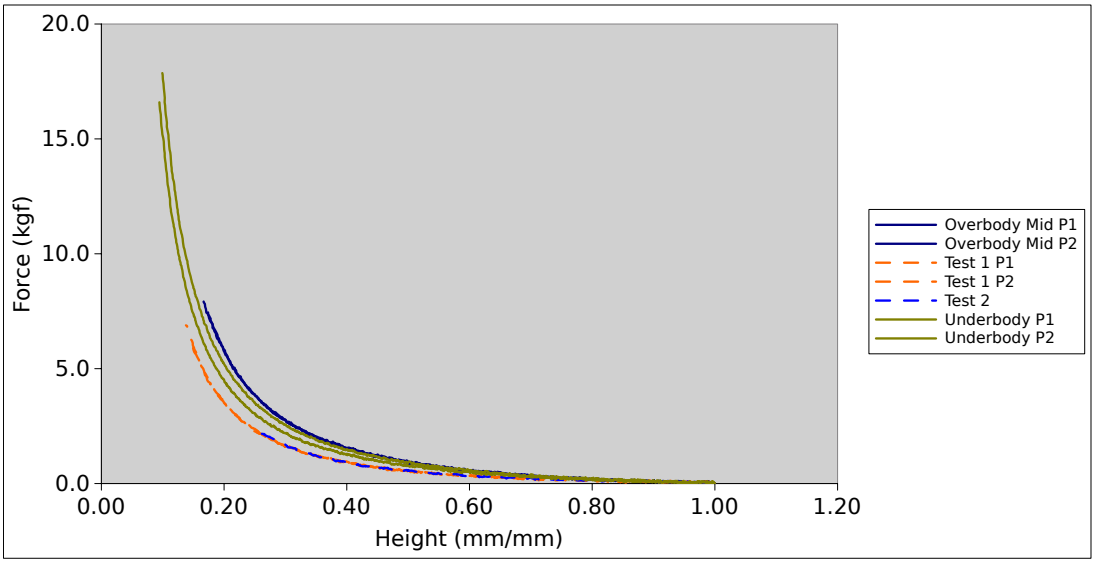
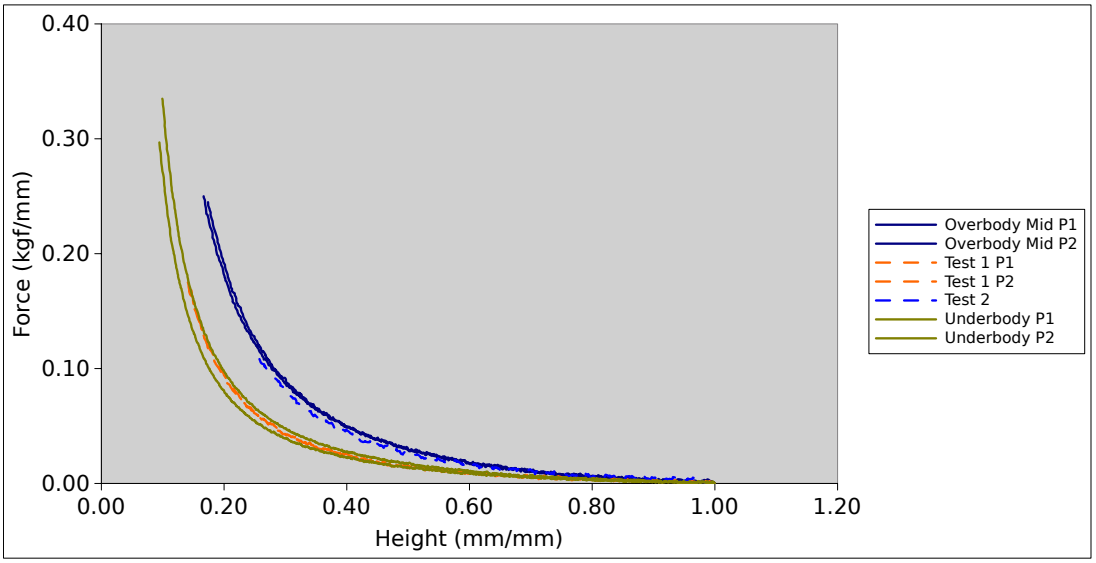


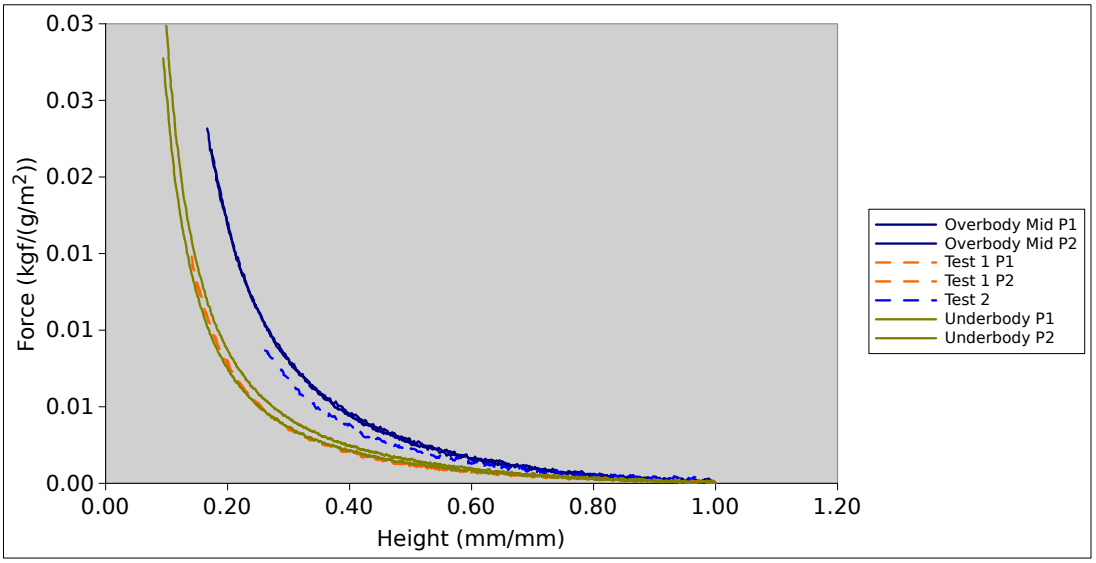
Figure 5.59: Force-displacement curves for the unbonded overbody and underbody knops.



(a) Force



(b) Force per thickness



(c) Force per weight

Figure 5.60: Force-displacement curves for overbody and underbody samples.

implying that the Test 2 sample is intrinsically more resistant (figs. 5.60b and 5.60c). Further adjustments to increase the weight for the overbody knoppy web retained this strength.

The general trend is that the thicker samples (Test 1 and the underbody sample) appear to be inherently less resilient to compression than the thinner samples (per figs. 5.60b and 5.60c). This is despite the fact that the underbody blend was specifically designed to be bulkier than the overbody blend.

5.6.4.3 Consumer trial of duvets

Table 5.25 summarises the results of the consumer trial. The best performance area was weight: participants were generally impressed at the warmth of the lightweight duvet, with nine of the participants stating that it was their favourite aspect of the duvet. It was also noted that there were no cold spots, which is consistent with the observation that loose fill duvets do not have any insulation along the quilted lines, while knoppy web duvets do. However, it was generally felt that the duvet was a spring/autumn product, with some participants finding it too hot during summer and others questioning whether it would be warm enough in winter. Possible solutions to this would be to have two weight targets, or to produce a lighter knoppy web for a summer product and layer it for a winter product.

Drape was the only performance area to garner a negative change (although still with an average score of 5.29/7). Participants noted that the duvet tended to sit on top of them rather than falling around them like a down and feather duvet. Three of the participants preferred this (saying that the duvet was not as constricting), but the majority found the feeling of stiffness detrimental.

5.6.5 Discussion

The most surprising and counter-intuitive result to come out of the product trials was that the overbody knoppy web blend was inherently more resilient to compression than the underbody blend (fig. 5.60), despite having specifically designed the underbody blend to be bulkier (section 5.6.1.2). This result is opposite to what was observed in the earlier knop resilience experiment (section 5.2.2). Two possible explanations spring to mind:

- The overbody knops were in fact more resilient.
- The DOA airlay lays thinner knoppy webs more densely.

The first explanation is consistent with the observed force-displacement curves for the overbody and underbody knops (fig. 5.59). The crimp of wool is known to have a significant effect on the compressional properties of bulk wool samples [108]. Even though the underbody blend used high-bulk wool, shorter finer wools can often have a relatively tight crimp [3], which could improve the bulk of the produced knops. By that reasoning, the overbody knops having a blend of mostly high-bulk and some short fine wool would end up bulkier than the underbody blend of equal high-bulk and coarser wool, as observed. Additionally, using a smaller knop means that there are more individual knops in the same weight, which could potentially store more strain energy than fewer larger knops, even if the smaller knops are softer.

However, the knops alone cannot explain fully explain the observed discrepancies. The airlayed weight and thickness appear to have more of an effect than the blend composition, because the first overbody test sample exhibited the same normalised compression profile as the underbody samples. While the exact airlay settings for the various samples cannot be provided here for commercial reasons, we can make two observations:

- The thickness of the sample is primarily set by the height of the exit roller (section 2.7), which defines the cross-sectional area of the exit outlet. A bulkier blend may "spring up" more once it clears the exit roller, but there is little stored strain energy with which to do so.

Table 5.25: Average rating change of the performance areas, and participant consensus [107]. The three consensus values represent the number of participants that rated the area higher (+), about the same (∼) or lower (-) after using the knoppy web duvet.

Performance area	Rating change	Consensus
Drape over body	–0.835	3+/3∼/6-
Weight	+1.33	8+/2∼/2-
Warmth	+0.5	5+/5∼/2-
Touch/feel	+0.36	5+/2∼/5-
Bedding stability	+0.33	4+/2∼/6-
Overall comfort	+0.875	7+/4∼/1-

- The bending moduli (intrinsic stiffness) of the overbody and underbody knoppy webs were negligibly different (table 5.23b). But if the thinner knoppy webs are indeed denser, the bending modulus would be expected to be higher. Weight is proportional to both density and thickness, and thus by eqs. (5.3) and (5.4), q is proportional to density. Thus it suggests that for the thinner samples, more of the web fibre is layed perpendicular to the direction of bending than for the thicker samples; fewer fibres parallel to the direction of bending would act to decrease bending strain.

The second of these observations is particularly interesting. It suggests that the drape of thinner knoppy webs could vary depending on whether bending stiffness is measured along or across the direction of airlaying. Intuitively this does make sense in general—the airlay lays fibre against an exit slot, so we can expect more fibres to be directed along the width or thickness of the airlayed batt than its length (at least for a regular fibre batt). Orientation was not a factor considered for the stiffness measurements in this thesis, as all bending samples were oriented across the airlay. A more in-depth study of the effect of orientation on drape would be beneficial, particularly for overbody and apparel knoppy web blends.

An alternative explanation could be that instead of the DOA airlay laying the thinner webs more densely, the thicker webs are more prone to pulling apart during transportation while unbonded. Recalling that the knoppy web rolls were bonded offsite (section 5.6.3.1), the higher weight of the underbody could have caused it to sag and stretch apart, effectively decreasing its density. However, this would not have affected the test samples, which were hand-cut and carefully transported flat to the onsite Moffat bonding oven.

The airlay density theory also provides an explanation for the observed disparity between the force-per-thickness and force-per-weight curves for varying knop:web ratio figs. 5.37b and 5.37c. Lot 3 was airlayed thinner and lighter than Lots 1 and 2 (table 5.8a); a higher density could explain why fig. 5.37c showed Lot 3 as having the highest compression resilience, while fig. 5.37b showed the expected relationship.

Regarding future product development, the implication of the above is that better mileage may be achieved by using two layers of a lower-weight underbody knoppy web in products, instead of a single higher-weight layer.

The issue of poor drape raised in the consumer trial (section 5.6.4.3) is intriguing. The flexural rigidity of the overbody knoppy web itself was not particularly high (table 5.23b), which suggests that the cause of the duvet stiffness is another factor:

- From measurements of the stiffness of downproof fabric in section 5.6.4.1, it was concluded that the fabric covering on products could inhibit their drape. The three fabrics used on the trial duvets were all lighter than downproof, and of the three the lightest fabric was qualitatively observed to offer the most drape. However it also showed signs of wear and tear at the end of the three-month trial. The effect of the fabric could be lessened by reducing the thickness of the knoppy web, which would also help to reduce the weight of the duvet.

- One of the aspects that the concurrent design-oriented study [106] examined was the effect of quilting. It was found that traditional grid quilting patterns were both inefficient (in the amount

of time and thread required) and detrimental to drape. The grid pattern is necessary for loose fills because they must be contained in pockets within the duvet, or else settling will occur. However, because the fill is loose, the duvet can bend along the quilting lines easily and drape well. For knoppy web duvets, the grid pattern has the effect of stiffening up the knoppy web because the quilting happens through the knoppy web instead of around it; novel stitch patterns can instead be used to improve the drape of the duvets.

The consumer trial report did not specify whether the stitching patterns of the duvets (section 5.6.3.2) had any correlation with the comments about drape. This is an avenue that should be investigated more closely in a future consumer trial, to confirm the stitch pattern findings [106].

Chapter 6

Conclusions and future work

6.1 Summary

The primary goal of this thesis was to aid the commercialisation of knoppy web by FibreTech New Zealand Ltd, both by conducting scientific research and contributing to the product development process. This two-pronged approach has resulted in two key contributions: optimised specifications for several types of knoppy web, and the first mathematical model describing the compression mechanics of a knoppy web.

To determine the optimised specifications, a series of experiments were conducted to examine key parts of the knoppy web production process. It was observed that knops can provide significantly better compressional properties than both down clusters and goose feathers (section 5.2), and that they can be packaged at high strain for extended periods and retain their desirable properties (after recovery via steaming) (section 5.3). Additionally, an investigation of the parameter space for knoppy web specifications (section 5.4) showed that there is significant tunability available in the properties of knoppy web. From these observations, optimum specifications were developed, that have been subsequently used by FibreTech New Zealand Ltd in the ongoing development of end user products (section 5.6).

In the model, a knoppy web is represented as a uniform array of knops embedded in a fibrous web. Each knop is treated as a hollow spherical membrane, to which a series of assumptions are applied such that the resulting sphere captures the core physical intuition of knop compression. The web is treated as a random fibrous assembly using van Wyk's equations. The model was developed using the energy method, and implemented using a variety of numerical techniques and computational packages.

The model was compared to knoppy web compression curve data taken during the investigation of the specification parameter space. Although its predictive powers are limited, the model was able to provide satisfactory first-order fits to the experimental data.

6.2 Validity of the hypotheses

In chapter 1 we proposed a series of hypotheses with regard to the benefits that the wool/PLA knoppy web product might provide over the industry standard wool batt. Restated in brief, the hypotheses were:

1. Knoppy web has higher bulk retention than plain wool batting.
2. Knoppy web has better drape than plain wool batting.
3. The presence of PLA in a product improves bulk retention.
4. The presence of PLA in a product improves its washability.

Hypothesis 1 is reasonably well-supported by the compression curves presented in section 5.4.3.7, and the corresponding model fits in section 5.5.6.2. While the model as-is cannot say anything about the effect of reduced fibre slippage, it is clear that the knoppy web curves are better-fitted by the knoppy web

model than web alone (tables 5.20 and 5.21), indicating that the knops do contribute to the observed behaviours. More specifically, the knops contribute more energy build-up at low strain (fig. 5.45). This does not immediately mean that knoppy web performs better under compression than plain wool batting, as no direct comparison was ever made. However, Lot 6 (where the knops were carded away) provides a close alternative to plain wool batting, and thus it is clear from fig. 5.39 that the knops provide a significant benefit.

Hypothesis 2 has much stronger experimental support, based on the stiffness comparisons (table 5.9a). While the compared Lots all had high knop percentages and were therefore at the far end of the scale from plain wool batting, the drastic change in intrinsic stiffness as the knop:web ratio was altered clearly showed that the general trend is easily consistent with this hypothesis. A stiffness comparison of Lot 2 to Lot 6 (table 5.9c) would appear to be contradictory to hypothesis 2, but it was reasoned that the change in effective PLA content of the web was the primary factor in that particular comparison.

Based on the compression curves in fig. 5.38, it would appear that hypothesis 3 is completely disproved, and the PLA content has no effect on the bulk retention of PLA. This would be a premature assertion. The experiment showed that over the considered range of PLA ratios, there was little to no effect; however, the experimental constraints meant that there was no data in the region between zero and 25% PLA content, which in hindsight is where any effect would most likely become visible. There is also the complicating factor of the PLA content within the knops, which was not a variable examined experimentally in any depth in this thesis. We therefore consider this particular hypothesis to be unproven.

On the other hand, there is specific evidence that contradicts hypothesis 4 (as discussed in section 5.4.4). It is certainly the weakest of the four hypotheses, and we consider it to be most likely invalid. To be fair though, the work in this thesis is insufficient to shut the door completely on this hypothesis. In particular, the precise phrasing of this hypothesis—that a product containing PLA is better than a product containing no PLA—means that it may still be viable. It is possible that the effect of PLA is only apparent at low fractions (e.g. the moment PLA fibres are present, fibre migration is inhibited per section 2.3.2.1), and increasing the PLA content higher has no additional effect. A more focused study of the effect on knoppy web of washing is required, looking at the region between zero and 25% PLA content.

6.3 Avenues for future research

6.3.1 Further development of the models

The theoretical models developed in this thesis rely on a series of assumptions to provide first-order approximations of the behaviour of knoppy web. Thus the most obvious future work would be to strip away some of these assumptions. Two in particular come to mind as candidates for removal:

- The assumption of uniform strain within the web component (as a consequence of assumption 4.3.3).
- The assumption that the knops are arranged in a uniform lattice (assumption 4.3.1).

As discussed in section 4.3.3, a proper non-uniform strain treatment of the web component would require either extending existing models of fibrous assemblies to non-uniform strain, or developing a new model of anisotropic fibrous assemblies distributed around spherical occlusions. Either of these would be considerable contributions. As an initial approximation however, one could extend the van Wyk energy method used here to remove the effects of assumption 4.3.3, perhaps by treating the web component as a series of columns or concentric cylinders. Such an approximation of course replaces existing assumptions with ones that are no better at describing the micromechanics within the web component. However, this would enable the model to generate a strain gradient within the web component more similar to fig. 4.34c, thereby potentially giving a better experimental fit than the modified knoppy web model.

The removal of assumption 4.3.1 would arguably give a more significant improvement to the model, particularly in regards to improving the physical relevance of the fitted parameters for the web component. In section 5.5.6.2 we pointed out that better fits were obtained as the percentage of web was increased and assumption 4.3.1 became more realistic. For desired knop:web ratios (such as those in the optimum specifications in section 5.6.2), the existing assumptions result in a web component that is much less dense than would be expected in reality (fig. 5.40). Packing the knops more closely would resolve this, but causes two further issues:

- The concept of a "unit cell" becomes more complex. If assumption 4.3.1 is relaxed inasmuch as the knops are not in a cubic matrix but remain in a uniform array, then a unit cell containing a single knop may remain viable. More likely, the unit cell would end up containing various fractions of multiple knops. Assuming that these fractions sum up to an integer number of knops, the knop model equations should require minimal adjustment; fractions of different knops that occupy the same logical position in the lattice could be modelled using the same knop simulation.

- The knop model is designed to handle uniaxial compression with a (cylindrical) radially-opposing force; its two geometric degrees of freedom (c and r') reflect this. In a more densely-packed structure, contact points with neighbouring knops would apply forces that physically wouldn't result in the same compression behaviour—the knop would be compressed with more than two degrees of freedom. It would certainly be possible however to calculate the energy of such a structure using the existing knop model, under the assumptions that the contact forces between knops are broken into components acting along the two geometric degrees of freedom. This would trade physical realism of the fitted web component parameters against physical realism of the knop compression.

6.3.2 Development of a knop model from a fibrous basis

The knop model developed in section 4.2 is obviously deficient in that it does not actually contain any fibrous components. The effect of this became particularly apparent while attempting to fit the model to knop data in section 5.5.4; it was necessary to add a fibrous component to account for the fibrous-like behaviour at high strains. A knop model developed from a fibrous basis would negate these issues; it could model the same physical behaviour at low strains as the current model, while at high strains the fibres would be in such close proximity that it would inherently behave more like a traditional fibrous assembly.

This effort would require significant new physics to be developed. As touched on in section 3.2, existing models of 2D and 3D fibrous assemblies can leverage the fact that the dimensions of the assembly are generally much larger than the fibre diameters. This is inherently false for a knop, and thus alternative strategies would need to be employed. Most likely, each fibre within a knop would need to be modelled individually, and their aggregate motion calculated under compression. A similar strategy was employed by Cassidy in determining the stress-strain properties of woollen yarns under tension [109]; each fibre was individually modelled as a series of fibre segments arranged in a helix.

One simplification that could initially be employed is to treat the fibres within a knop as continuous hoops. This would require the assumption that the length of the fibres greatly exceeds the circumference of the knop (which is reasonable enough for smaller knops), and is similar to assumption 4.2.11 employed in our model. The knop would then consist of a randomly-oriented set of hoops, which would also need to be randomly interlocked in some fashion. This situation would simplify the stress-strain calculations for the fibre tension, while fibre bending could be determined based on the deformation of individual segments of the hoop.

6.3.3 Model of a partially-bonded fibrous assembly

Another deficiency of the current model is that it does not account for the PLA-PLA bonds behaving differently to other contact points; it simply treats the PLA fraction as a stiffening term and fits the model parameters appropriately. Putting aside the issue of PLA-PLA bonds within the knop, the web component bonds could be handled simply by picking a more suitable model. However, as pointed out in section 5.5.1.4, none of the models of fibrous assemblies considered in section 3.3 have any concept of distinct types of contact points. Most assume that all contact points are bonded or fixed, and all assume that all contact points behave in the same way.

In order to properly account for the presence of bonded PLA, it would be necessary to extend one of the models with a second contact point type. The most likely candidate would be Carnaby and Pan's model [14] (covered in section 3.3.2.3), where a slipping contact point was considered. Adding a permanently-fixed contact point should be possible by preventing a certain fraction of contact points from being able to slip. This fraction could then be estimated from the number ratio of fibre types, which was linked to their mass ratio in section 2.5.5. Incorporating this model into our knoppy web model would of course require additional work, both to convert the Carnaby Pan model to the energy method, and to alter the current modelling strategy to account for the iterative nature of the contact point force calculations.

6.3.4 The effect of wool blends on knops

In determining the scope of this thesis, knops were considered a known quantity. The technology had been under development for several decades, and the production parameters were seemingly well-known; thus the focus was on the use of knops in products, rather than their creation (per section 2.6). However, in the development of optimal knoppy web specifications (section 5.6), the fibre blend became the primary distinguishing factor over other properties such as knop:web ratio; this indicates that in the face of user requirements, the fibre blend needs to be carefully understood. We also established in section 5.5.1.2 that the fibre blend within a knop has an effect on how the production parameters can be modelled.

A useful future project would therefore be to focus on knop development, and investigate the effect of different fibre blends on knop formation and properties. At a high level, this could involve quantitiatively varying the fibre blend along various axes (e.g. source fibre length distribution, fibre diameter, cutting length, crimp) and measuring the resulting properties after processing with a fixed knopping specification. This would treat the proprietary knopping process as a "black box." A more in-depth study would require delving into the trade secrets at FibreTech New Zealand Ltd to elucidate the actual micromechanical knopping process, from which a broader understanding of the effect of wool blends could be obtained.

6.3.5 Theory of washing

This primary theoretical focus of this thesis was on the behaviour under compression of knoppy web, but equally important is understanding its behaviour subject to washing. More generally, while there is some existing literature on the felting of wool fibres (e.g. [110, 111]) and the effects of washing on fabrics (e.g. [112, 113]), the problem of modelling the behaviour of non-woven fibre batting under washing is essentially unstudied.

In section 2.3.2 we discussed the withdrawal force of a fibre from a tuft, and pointed out that the work of Lee et al. [34] could form the basis of a theory describing the micromechanics of washing fibrous assemblies. Developing upon this idea would be a valuable contribution to both academia and industry; moreover, it would provide a useful stepping stone towards developing a model describing the behaviour of knoppy web under washing. A model that incorporated knops and PLA alongside wool could help to explain the issues surrounding hypothesis 4 (discussed in section 5.4.4).

References

[1] Ensminger, M. E. and Parker, R. O. *Sheep & Goat Science*. Interstate Printers & Publishers, Danville, Illinois, 5th edition, 1986. 1

[2] Lenzing. **The Global Fiber Market in 2014** [online]. 2015. Available from: `http://www.lenzing.com/en/investors/equity-story/global-fiber-market.html` [cited 2015-06-02]. 1

[3] Fraser, R. D. B. and Rogers, G. E. **The bilateral structure of wool cortex and its relation to crimp**. *Australian Journal of Biological Sciences*, **8**:288–299, 1955. 1, 186

[4] Munro, W. A. and Carnaby, G. A. **Wool-fibre Crimp Part I: The Effects of Microfibrillar Geometry**. *Journal of the Textile Institute*, **90**(2):123–136, 1999. 1

[5] American Sheep Industry Association. **Characteristics of Wool Fact Sheet**. 1

[6] Plante, A. M., Holcombe, B. V., and Stephens, L. G. **Fiber Hygroscopicity and Perceptions of Dampness: Part I: Subjective Trials**. *Textile Research Journal*, **65**(5):293–298, 1995. 1

[7] Feughelman, M. **A Note on the Recoverability of Mechanical Properties in Wool Fibres**. *Journal of The Textile Institute*, **59**(11):548–550, 1968. 1, 113

[8] Chapman, B. M. **The relationship between single fibre bending behavour and fabric wrinkle recovery**. In *Proceedings of the 5th International Wool Textile Research Conference*, **3**, pages 483–492, Aachen, Germany, 1975. 1, 113, 129

[9] Postle, R., Carnaby, G. A., and de Jong, S. **The viscoelastic properties of the wool fibre**. In *The Mechanics of Wool Structures*, chapter 2, pages 35–54. 1988. 1, 113

[10] Ingham, P. E., Edwards, R. J., and Youngman, P. **The flammability of apparel fabrics sold in New Zealand**. Technical report, Christchurch, New Zealand, 1983. 1

[11] Johnson, N. A., Wood, E. J., Ingham, P. E., McNeil, S. J., and McFarlane, I. D. **Wool as a Technical Fibre**. *Journal of the Textile Institute*, **94**(3-4):26–41, 2003. 1

[12] Alongi, J., Horrocks, A. R., and Malucelli, G. **Fundamental Aspects of Flame Retardancy**. In *Update on Flame Retardant Textiles: State of the Art, Environmental Issues and Innovative Solutions*, chapter 2. Smithers Rapra, Shrewsbury, Shropshire, GBR, 2013. 1

[13] Dunlop, J. I. **On the Compression Characteristics of Fibre Masses**. *Journal of the Textile Institute*, **74**:92–97, 1983. 3

[14] Carnaby, G. A. and Pan, N. **Theory of the Compression Hysteresis of Fibrous Assemblies**. *Textile Research Journal*, **59**(5):275–284, 1989. xiii, 3, 52, 54, 159, 192

[15] Hearle, J. W. S. **A Qualitative View of Spun Yarn Mechanics - A Modified Qualitative Approach**. In *Structural Mechanics of Fibers, Yarns and Fabrics - Volume 1*, chapter 7, pages 277–283. Wiley-Interscience, 1969. 3

[16] Postle, R., Carnaby, G. A., and de Jong, S. **The mechanics of wool carpets**. In *The Mechanics of Wool Structures*, chapter 16, pages 411–435. Ellis Horwood, 1988. 3

[17] McPhee, J. R. **Shrink-Resist Treatment and Laundering of Wool Fabrics**. *Textile Research Journal*, **31**(12):1045–1053, 1961. 3

[18] Makinson, K. R. *Shrinkproofing of Wool*. Fiber science series. Marcel Dekker, 1979.

[19] Holme, I. **New Developments in the Chemical Finishing of Textiles**. *Journal of the Textile Institute*, **84**(4):520–533, 1993. 3

[20] Gao, J., Yu, W., and Pan, N. **Structures and Properties of the Goose Down as a Material for Thermal Insulation**. *Textile Research Journal*, **77**(8):617–626, 2007. 3, 112

[21] Wilde, T. P., McDowell, D. L., Jacob, K. I., and Aneja, A. P. **A Modified Mullins Model for Compressive Behavior of Goose Down Fiber Assemblies**. *Mechanics of Advanced Materials and Structures*, **13**(1):83–93, 2006. 3

[22] Crow, R. M., Nolan, R. W., Cattroll, S. W., and Dewar, M. M. **Comparison of Down- and Feathers and Synthetic Insulants for Use in Sleeping Bags**. Technical report, Defence Research Establishment, Ottawa, Ontario, 1982. 3

[23] Watzl, A. **Fusion bonding thermobonding and heat setting of nonwovens - theoretical fundamentals, practical experience, market trends. [I]**. *Melliand Textilberichte*, **75**(10):840–850 + E217, 1994. 5

[24] Hoffmeyer, F. J. and Watt, J. D. **Thermo-bonding of lofty wool batts with low-melting-temperature fibres**. Technical report, Christchurch, N.Z., 1986. 5

[25] Tokiwa, Y. and Jarerat, A. **Biodegradation of poly(L-lactide)**. *Biotechnology Letters*, **26**(10):771–777, 2004. 5

[26] Royte, E. **Corn Plastic to the Rescue**. *Smithsonian Magazine*, 2006. 5

[27] Coszach, P. and Willocq, J. **Method for stereospecifically recycling a PLA polymer mixture**, 2010. 5

[28] Lawrence, C. A. *Fundamentals of Spun Yarn Technology*. CRC Press, 2003. xiii, 11, 20, 21, 31

[29] Martindale, J. G. **The Distribution and Movement of Wool on Woollen Cards**. *Journal of the Textile Institute*, **36**(9):T213–T228, 1945. 12

[30] Monfort, F. **Carding as a Markovian Process**. *Journal of the Textile Institute*, **53**(8):T379–T393, 1962. xiii, 13, 17

[31] Grosberg, P. **The Strength of Twistless Slivers**. *Journal of the Textile Institute*, **54**:T223–233, 1963. 15

[32] Grosberg, P. and Smith, P. A. **The Strength of Slivers of Relatively Low Twist**. *Journal of the Textile Institute*, **57**:T15–T23, 1966. 15

[33] Johnson, K. L. **Rolling Contact of Elastic Bodies**. In *Contact Mechanics*, chapter 8. Cambridge University Press, 1987. 15

[34] Lee, M. E. M., Kozyreff, G., Howell, P. D., and Ockendon, H. **A Model for the Break-up of a Tuft of Fibers**. *Physical Review E*, **74**(4):1–10, 2006. xiii, 16, 17, 18, 192

[35] Carnaby, G. A. and Burling-Claridge, G. R. **Carding of Tender Wool: Part I: Theory**. *Textile Research Journal*, **66**(2):90–98, 1996. 18, 19

[36] Carnaby, G. A. **Fiber Breakage During Carding Part I: Theory**. *Textile Research Journal*, pages 366–369, 1984. 18

[37] Wood, E. J., Stanley-Boden, P., and Carnaby, G. A. **Fiber Breakage During Carding Part II: Evaluation**. *Textile Research Journal*, pages 419–424, 1984. 18

[38] Meyer, R., Almin, K. E., and Steenberg, B. **Length reduction of fibres subject to breakage**. *British Journal of Applied Physics*, **17**(3):409, 1966. 18, 20

[39] Burling-Claridge, G. R. **Carding of Tender Wool: Part II: Evaluation of the Model**. *Textile Research Journal*, **66**(3):141–150, 1996. 18

[40] SGS Wool Testing Services. **Fibre Length**. Technical report, SGS, Wellington, New Zealand, 2011. 18

[41] International Wool Textile Organisation. **IWTO-17-85: Determination of fibre length distribution parameters by means of the almeter**. Technical report, International Wool Textile Organisation, 1985. 18

[42] Perel, J. **Characterization of Fiber Length in Slivers**. *Textile Research Journal*, **52**(6):376–379, 1982. 18

[43] Grignet, J. **Foundations and Metrology of the Almeter Wool Fibre Length Measurement**. Technical report, International Wool Textile Organisation, Istanbul, 2003. 18

[44] Harrowfield, B. V. **The Systematic Cutting and Stretch-Breaking of Disoriented Slivers**. *Textile Research Journal*, **49**(10):565–575, 1979. xiii, 20, 21, 23

[45] Lindsley, C. H. **Measurement of Fiber Orientation**. *Textile Research Journal*, **21**(1):39–46, 1951. 20, 21, 24

[46] Tallant, J. D. and Pittman, R. A. **Breakage Models and Measuring Techniques for Fiber. Weight-Length Distributions**. *Textile Research Journal*, **38**(2):149–155, 1968. 20

[47] Burling-Claridge, G. R. and Carnaby, G. A. **Cutting A Sliver**. In *8th International Wool Textile Research Conference*, pages III.286–III.294, 1990. xiii, 20, 22

[48] Morton, W. E. and Summers, R. J. **Fibre Arrangement in Card Slivers**. *Journal of the Textile Institute*, **40**:106–116, 1949. 24

[49] Wakankar, V. A., Bhaduri, S. N., Ramaswamy, B. R., and Ghosh, G. C. **Some Studies on the Formation of Hooks in Carding**. *Textile Research Journal*, **31**(11):931–940, 1961. 24

[50] Neckář, B. and Ibrahim, S. *Structural theory of fibrous assemblies and yarns: Structure of fibrous assemblies*. Technical University, 2003. 24

[51] Lawrence, C. A. **Materials Preparation Stage III: Drawing, Combing, Tow-Top Conversion, Roving Production**. In *Fundamentals of Spun Yarn Technology*, chapter 5. 2003. 30

[52] Martindale, J. G. **4—A NEW METHOD OF MEASURING THE IRREGULARITY OF YARNS WITH SOME OBSERVATIONS ON THE ORIGIN OF IRREGULARITIES IN WORSTED SLIVERS AND YARNS**. *Journal of the Textile Institute Transactions*, **36**(3):T35–T47, 1945. 30

[53] Matbase. **PLA monomere (Polylactic Acid)** [online]. 2015. Available from: `http://www.matbase.com/material/polymers/agrobased/polylactic-acid-pla/properties` [cited 2015-10-26]. 33

[54] Townend, P. P., Hewitt, A., and Chu, C. K. **COMPARATIVE CONTRIBUTION OF SOME CARD VARIABLES TO THE FORMATION OF NEPS, USING METALLIC CLOTHING**. *TEXTILE INSTITUTE AND INDUSTRY*, **17**(3):100–102, 1979. 33

[55] Harrison, R. E. and Bargeron, J. D. **Comparison of Several Nep Determination Methods'**. *Textile Research Journal*, **56**(2):77–79, 1986. 33

[56] Russell, S., editor. *Handbook of Nonwovens*. Woodhead Publishing, 2006. xiii, 33, 35

[57] Howitt, F. O. **12—THE YELLOWING OF WOOL: A SURVEY OF THE LITERATURE**. *Journal of the Textile Institute Transactions*, **55**(2):T136–T145, 1964. 37

[58] Komori, T. and Makishima, K. **Numbers of Fiber-to-Fiber Contacts in General Fiber Assemblies**. *Textile Research Journal*, **47**(1):13–17, 1977. xiii, 41, 49, 51

[59] Gregory, R. D., Milac, T. I., and Wan, F. Y. M. **A Thick Hollow Sphere Compressed by Equal and Opposite Concentrated Axial Loads: An Asymptotic Solution**. *SIAM Journal on Applied Mathematics*, **59**(3):1080–1097, 1998. 42

[60] Pauchard, L. and Rica, S. **Contact and compression of elastic spherical shells: the physics of a 'ping-pong' ball**. *Philosophical Magazine B*, **78**(2):225–233, 1998. 42

[61] Shorter, R., Smith, J. D., Coveney, V. A., and Busfield, J. J. C. **Axial compression of hollow elastic spheres**. *Journal of Mechanics of Materials and Structures*, **5**(5), 2010. 42

[62] Cross, R. **Dynamic properties of tennis balls**. *Sports Engineering*, (2):23–33, 1999. 42

[63] Sternberg, E. and Rosenthal, F. **The elastic sphere under concentrated loads**. *Journal of Applied Mechanics*, **19**:413–421, 1952. 42

[64] Guerrero, I. and Turteltaub, M. J. **The elastic sphere under arbitrary concentrated surface loads**. *Journal of Elasticity*, **2**(1):21–33, 1972. 42

[65] Lin, Y.-L., Wang, D.-M., Lu, W.-M., Lin, Y.-S., and Tung, K.-L. **Compression and deformation of soft spherical particles**. *Chemical Engineering Science*, **63**(1):195–203, 2008. 42

[66] Rawal, A., Mishra, P. K., and Saraswat, H. **Modeling the compression-induced morphological behavior of nonwoven materials**. *Journal of Materials Science*, **47**(5):2365–2374, 2012. 43

[67] van Wyk, C. M. **20—NOTE ON THE COMPRESSIBILITY OF WOOL**. *Journal of the Textile Institute Transactions*, **37**(12):T285–T292, 1946. 44, 45, 48, 49, 57, 59

[68] Eggert, M. and Eggert, J. **No Title**. In Mark, H., editor, *Beitrage zur Kenntnis der Wolle und Ihrer Bearbeitung*, pages 69–87. Gebruder Borntraeger, Berlin, Germany, 1925. 44

[69] Lee, D. H., Carnaby, G. A., Carr, A. J., and Moss, P. J. **A review of current micromechanical models of the unit fibrous cell**. *Wool Research Organisation of New Zealand Communications*, 1990. 49

[70] Lee, D. H. and Lee, J. K. **Initial compressional behaviour of fibre assembly**. *Objective Measurement: Applications to Product Design and Process Control*, 1985. xiii, 50, 51, 52, 72, 76

[71] Lee, D. H. and Carnaby, G. A. **Compressional Energy of the Random Fiber Assembly - Part I: Theory**. *Textile Research Journal*, **62**(4):185–191, 1992. xvi, 53, 201, 202

[72] Lee, D. H., Carnaby, G. A., and Tandon, S. K. **Compressional Energy of the Random Fiber Assembly - Part II: Evaluation**. *Textile Research Journal*, **62**(5):258–265, 1992. 53, 72, 83, 159, 166, 173, 201, 206, 207, 208, 209, 210, 212, 284

[73] Pan, N. **A Modified Analysis of the Microstructural Characteristics of General Fiber Assemblies**. *Textile Research Journal*, **63**(6):336–345, 1993. 53, 54

[74] Komori, T. and Itoh, M. **A Modified Theory of Fiber Contact in General Fiber Assemblies**. *Textile Research Journal*, **64**(9):519–528, 1994. 53, 54, 55

[75] Picu, R. C. **Mechanics of random fiber networks - a review**. *Soft Matter*, **7**(15):6768–6785, 2011. 55

[76] Bower, A. F. *Applied Mechanics of Solids*. CRC Press, 1st edition, 2009. 60

[77] Hamidzadeh, H. R. and Jazar, R. N. **Governing Equations**. In *Vibrations of Thick Cylindrical Structures*, chapter 2. 2010. 63

[78] Timoshenko, S. and Goodier, J. N. *Theory of Elasticity*. McGraw-Hill, 3rd edition, 2001. 66

[79] Komori, T., Itoh, M., and Takaku, A. **A Model Analysis of the Compressibility of Fiber Assemblies**. *Textile Research Journal*, **62**(10):567–574, 1992. 72

[80] The Engineering ToolBox. **Modulus of Elasticity or Young's Modulus - and Tensile Modulus for some common Materials** [online]. Available from: `http://www.engineeringtoolbox.com/young-modulus-d_417.html` [cited 2013-02-11]. 72

[81] Anderson, M. L., Mott, P. H., and Roland, C. M. **The Compression of Bonded Rubber Disks**. *Rubber Chemistry and Technology*, **77**(2):293–302, 2004. 72

[82] Python Software Foundation. **Python**. 72, 83, 206

[83] Pérez, F. and Granger, B. E. **IPython: a System for Interactive Scientific Computing**. *Computing in Science and Engineering*, **9**(3):21–29, 2007. 72, 206

[84] SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2014. 72, 206

[85] van der Walt, S., Colbert, S. C., and Varoquaux, G. **The NumPy Array: A Structure for Efficient Numerical Computation**. *Computing in Science & Engineering*, **13**(2):22–30, 2011. 72, 206

[86] Jones, E., Oliphant, T., Peterson, P., and Others. **SciPy: Open source scientific tools for Python**, 2001. 72, 166, 206

[87] The Woolmark Company. **TM 272: Low Pressure Bulk of Loose Wool Fills (For Over-Body Bedding & Pillows)**. Technical report, 2000. 109, 110, 113, 116, 119, 131, 135, 156

[88] European Committee for Standardization. **EN 12130:1998 Feather and down - Test methods - Determination of the filling power (massic volume)**, 1998. 110

[89] IDFL. **Evaluation of Fill Power Conditioning Methods**. Technical report, 2011. 112

[90] Bedford, J., Ross, D. A., Carnaby, G. A., and Lappage, J. **WRONZ fibre bulkometer instruction manual**. Technical report, Wool Research Organisation of New Zealand, 1977. 119, 131

[91] Instron Corporation. **MODEL 4204 LOADING FRAME DESCRIPTION AND OPERATING INSTRUCTIONS**. Technical report, 1990. 119, 135

[92] WeatherOnline Ltd. **Max temperature Christchurch - Observations 09.2015** [online]. 2015. Available from: `http://www.weatheronline.co.nz/weather/maps/city?WMO=93781&ART=MAX&LEVEL=150&MM=09&YY=2015&WEEK=2` [cited 2015-10-11]. 125

[93] WeatherOnline Ltd. **Relative humidity Christchurch - Observations 09.2015** [online]. 2015. Available from: `http://www.weatheronline.co.nz/weather/maps/city?WMO=93781&ART=RLF&LEVEL=150&MM=09&YY=2015&WEEK=2` [cited 2015-10-11]. 125

[94] Speakman, J. B. **The rigidity of wool and its change with adsorption of water vapour**. *Transactions of the Faraday Society*, **25**:92–103, 1929. 125

[95] Wilson, C., Dunn, L., and Laing, R. **Thermal properties of duvets with differing knop:web and PL51:wool ratios**. Technical report, Clothing and Textiles Centre, University of Otago, 2013. xvii, 132, 144, 145

[96] International Organization for Standardization. **ISO 11092: Textiles - Physiological effects - Measurement of thermal and water-vapour resistance under steady-state conditions (sweating guarded-hotplate test)**, 1993. 132

[97] National Fire Protection Association. **NFPA: 1971 Protective clothing for structural fire fighting - Appendix B**, 1990. 132

[98] British Standards Institution. **BS 3356:1961 Method for the determination of stiffness of cloth**, 1961. 133

[99] The Woolmark Company. **TM 274: Entanglement and Felting of Loose Soft Knop Fills For Bedding**. Technical report, 2000. 135

[100] Song, C., Wang, P., and Makse, H. A. **A phase diagram for jammed matter**. *Nature*, **453**(7195):629–632, 2008. 156

[101] Munroe, R. **Experiment** [online]. 2009. Available from: `https://xkcd.com/669/` [cited 2016-02-09]. 157

[102] Newville, M., Stensitzki, T., Allen, D. B., and Ingargiola, A. **LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python**, 2014. 161, 166

[103] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. **A Limited Memory Algorithm for Bound Constrained Optimization**. *SIAM Journal on Scientific Computing*, **16**(5):1190–1208, 1995. 166

[104] Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. **Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization**. *ACM Transactions on Mathematical Software*, **23**(4):550–560, 1997. 166

[105] Kaw, A. K. *Mechanics of Composite Materials*. Mechanical and Aerospace Engineering Series. CRC Press, 2nd edition, 2005. 175

[106] Olatunji, K. R. *Capturing Loft: Adding value to New Zealand wool bedding products through textile design innovation*. Master's exegesis, Massey University, 2014. 176, 187, 188

[107] Harman, R. **Delta Duvet Consumer Trials Report**. Technical report, FibreTech NZ Ltd., 2015. xvii, 181, 187

[108] Chaudri, M. A. and Whiteley, K. J. **The Influence of Natural Variations in Fiber Properties on the Bulk Compression of Wool**. *Textile Research Journal*, **38**(9):897–906, 1968. 186

[109] Cassidy, B. D. *Simulating the Stress-Strain Properties of Woollen Yarns*. PhD thesis, Lincoln University, 1997. 191

[110] Stigter, D. **On a correlation between the surface chemistry and the felting behavior of wool**. *Journal of the American Oil Chemists' Society*, **48**(7):340–343, 1971. 192

[111] Stigter, D. **Wool Felting and Electrostatic Interaction Between Wool Fibers in Salt and Detergent Solutions**. *Textile Research Journal*, **42**(11):657–660, 1972. 192

[112] Bogaty, H., Lourigan, G. H., and Harris, H. E. **Structural Compactness of Woven Wool Fabrics and Their Behavior in Modern Washing Machines**. *Textile Research Journal*, **28**(9):733–737, 1958. 192

[113] Makinson, K. R. **Studies of the Movement of Wool Fibers in Fabrics During Felting, With Particular Reference to the Permanency of Pleats: Part I: Light Felting and its Effect on Pleats in a Worsted Fabric**. *Textile Research Journal*, **29**(5):431–439, 1959. 192

[114] Lee, D. H., Carnaby, G. A., and Tandon, S. K. **An energy analysis of the compression of a random fibre assembly**. *Proceedings of the 8th International Wool Textile Conference*, **V**:34–44, 1990. 203

Appendix A

The Lee Carnaby model

A.1 Overview

Lee and Carnaby [71, 72] developed a new micromechanical model for uniaxial compression of a fibre assembly using the energy method. The motivation behind this new model is unclear in the paper, but it brings together several novel and interesting concepts. At an earlier stage in this thesis we attempted to use this model to represent the web component of knoppy web; in doing so we discovered a previously-unmentioned flaw in the model that significantly limits its range of applicability.

In this appendix, we outline the key elements of the Lee Carnaby model. We re-implement the model using modern computational methods, and recreate key figures [72] to prove the correctness of the implementation. Finally, we examine the flaw in the model that prevents it from being used in a knoppy web model.

A.2 Model description

Figure A.1 shows the bending element for this model. A curved bending element is used, and its direction is given by the spherical polar coordinates of the chord of the element. In this model, the micromechanical action is for the curvature of the bending element to change—and it is allowed to either increase or decrease. This differs from the straight-element models, where only deflection of the beam was considered (ie. an increase in curvature). It is assumed that all bending elements start with the same curvature, that the curvature after compression is uniform, and that the bending elements have no torsion. While not explicitly stated in the paper, the no-torsion assumption combined with the given geometry implies that the planes of curvature of all bending elements are aligned with the vertical axis.

Let the curvature of the ith bending element before compression be κ_{1i}, and the curvature after compression be κ_{2i}. Geometrically, the relationship between the length of the bending element l_{1i} and the chord length of the bending element b_{1i} before compression is

$$b_{1i} = \frac{2}{\kappa_{1i}} \sin \frac{l_{1i}\kappa_{1i}}{2}, \tag{A.1}$$

and similarly after compression of the bending element for the relationship between l_{2i} and b_{2i}.

Lee and Carnaby continue with the assumption of affine deformation that previous workers have assumed. Consequentially, they define the compressional strain along the vertical axis ϵ_a using the standard strain equation as

$$\epsilon_a = \frac{z_{2i} - z_{1i}}{z_{1i}}, \tag{A.2}$$

where z_{1i} and z_{2i} are the projections onto the z axis of b_{1i} and b_{2i} respectively. Similarly, they define the

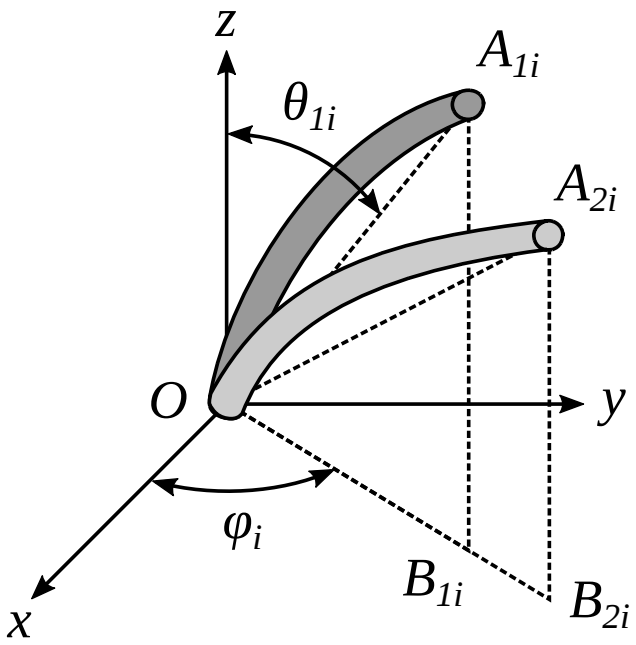


Figure A.1: Configurations of a fibre segment before (OA_{1i}) and after (OA_{2i}) z-directional compression, and their projected chords OB_{1i} and OB_{2i} on the xy plane [71].

Poisson's ratio of the fibre assembly ν_a as

$$\begin{aligned}\nu_a &= -\frac{\frac{x_{2i}-x_{1i}}{x_{1i}}}{\epsilon_a} \\ &= -\frac{\frac{y_{2i}-y_{1i}}{y_{1i}}}{\epsilon_a},\end{aligned} \tag{A.3}$$

using the projections of the chord lengths onto the x and y axes. This equation follows from the observation that an initially-random fibre assembly will become transversely isotropic upon compression.

While the uncompressed fibre assembly will have a single Poisson's ratio (under the randomness assumption), under compression it becomes another variable that needs to be accounted for. Lee and Carnaby solve this issue in the same manner as for their earlier model [114]. The minimum energy principle states that a system can be in elastic equilibrium only when the total energy of the system is at a minimum. Having assumed that all fibres are linearly elastic, Lee and Carnaby define that the true Poisson's ratio at any particular compressional strain is simply the value for which the total energy of the system is minimized. Therefore, by making ϵ_a and ν_a the independent variables in their model, they can fix ϵ_a and then calculate the energy of the assembly for a range of values of ν_a, with the accuracy of the fit only limited by computing power.

Upon performing this change of variables (and also replacing the discrete variables with their continuous counterparts), the following relationship is obtained:

$$b_2 = b_2 g(\theta_1), \tag{A.4}$$

where

$$g(\theta_1) = \cos^2\theta_1(1+\epsilon_a)^2 + \sin^2\theta_1(1-\nu_a\epsilon_a)^{2^{1/2}} \tag{A.5}$$

and θ_1 is the polar angle of the direction of the chord of the element before compression.

It is clear from this relationship that the condition $g(\theta_1) = 1$ denotes a critical boundary at which point the bending element can be compressed without a change of curvature. This is equivalent to a critical angle θ_c which satisfies that condition,

$$\theta_c = \sin^{-1}\left[\frac{2+\epsilon_a}{(1+\nu_a)(2+[1-\nu_a]\epsilon_a)}\right]^{1/2}. \tag{A.6}$$

Thus, a fibre element will straighten if its initial polar angle is greater than θ_c, and will bend if its initial polar angle is less than θ_c.

It is worth noting that θ_c has an explicit dependence on both the compressional strain and the Poisson's ratio. Figure A.2 shows a surface plot of θ_c, from which it is apparent that θ_c decreases as ϵ_a becomes more negative (higher compressional strain), and as ν_a increases. Functionally, this means that during continuous compression, fibre segments that were bending segments can transition to straightening segments—or vice versa, depending on the values to which ν_a is minimised.

Putting aside the compression behaviour of θ_c,

$$\kappa_2^2 = 6\left(\frac{2}{l_2}\right)^2\left(1-\frac{b_1 g(\theta_1)}{l_2}\right). \tag{A.7}$$

Assuming that the fibre segment length does not change under deformation (ie. $l_2 = l_1$), a fibre segment with polar angle less than θ_c can be bent to an arbitrary curvature without limit, because $(b_1 g(\theta_1))/(l_2) < 1$ and the equation is well-behaved. For a straightening segment, this is not the case. The segment will straighten until the curvature is zero, where $(b_1 g(\theta_1))/(l_2) = 1$. Beyond this point, $(b_1 g(\theta_1))/(l_2) > 1$ and there is no real value for κ_2.

To avoid this inconsistency, the assumption that the fibre segment length does not change must be modified to allow l_2 to increase once the fibre segment has fully straightened, in such a way that $\kappa_2 = 0$. There are two physical mechanisms that could lead to this: extension, and slippage. Lee and Carnaby

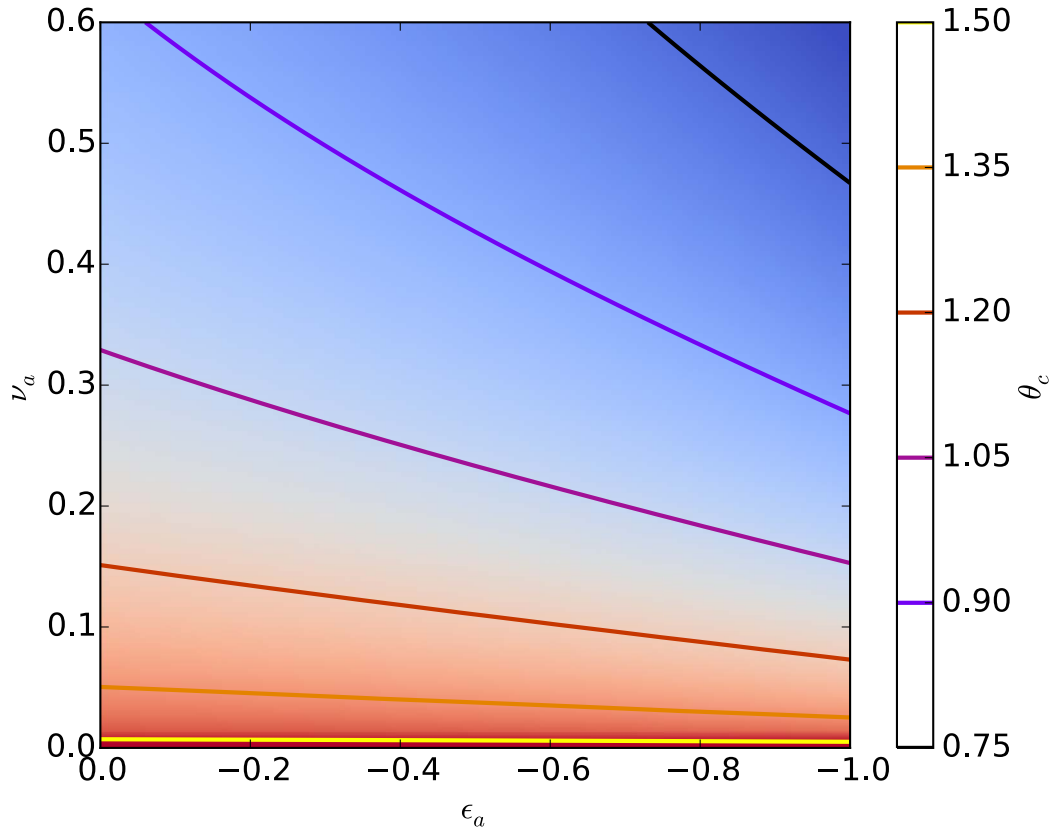


Figure A.2: Contour plot of θ_c.

surmised that fibre extension occurs very rarely; the friction of the contact points is not so great that it could withstand lateral tension. Therefore, they suggest that slippage is the mechanism by which tension is avoided.

The boundary condition for whether a fibre element will be straightening or slipping can be obtained from eq. (A.7). Defining l_c to be the value of l_1 for which $\kappa_2 = 0$,

$$l_c = \frac{1}{\kappa_1}\left\{24\left(1 - \frac{1}{g(\theta_1)}\right)\right\}^{1/2}. \tag{A.8}$$

Following this analysis of the bending element, the next stage of the model is counting the number of contacts. Sections 3.3.1.2 and 3.3.2.1 already detail the equations for determining the total number of contacts in a fibre assembly N, but this count must be further divided into the number of bending, straightening and slipping elements. Because a fibre element can only be one of the types at any particular stage,

$$N = N_B + N_T + N_S, \tag{A.9}$$

where N_B, N_T and N_S are the number of bending, straightening and slipping elements respectively.

Lee and Carnaby derive these counts by way of a joint probability distribution function over the fibre elements. The three variables that define an individual fibre element are its length, orientation and curvature. The latter of these is assumed constant, and under the random distribution assumption the length and orientation are independent, so

$$J(l, \theta) = f(l)\Omega(\theta)\sin\theta, \tag{A.10}$$

where $f(l)$ is the fibre element length density function, and $\Omega(\theta)\sin\theta$ is the orientation density function. From the definition of probability density,

$$\int_0^{\pi/2}\int_0^{\infty} J(l, \theta) dl d\theta = 1. \tag{A.11}$$

Recalling that θ_c and l_c are the boundaries between the various element types, we can simply write the number of each element type as

$$N_B = N\int_0^{\theta_c}\int_0^{\infty} J(l_1, \theta_1) dl_1 d\theta_1 \tag{A.12}$$

$$N_T = N\int_{\theta_c}^{\pi/2}\int_0^{l_c} J(l_1, \theta_1) dl_1 d\theta_1 \tag{A.13}$$

$$N_S = N\int_{\theta_c}^{\pi/2}\int_{l_c}^{\infty} J(l_1, \theta_1) dl_1 d\theta_1. \tag{A.14}$$

From the perspective of reducing computational complexity, it is useful to note that N_B can be reduced further because l_1 and θ_1 are assumed independent:

$$N_B = N\int_0^{\theta_c} \Omega(\theta)\sin\theta d\theta_1 \int_0^{\infty} f(l) dl_1 \tag{A.15}$$

$$= N\int_0^{\theta_c} \Omega(\theta)\sin\theta d\theta_1. \tag{A.16}$$

This reduction is not possible for N_T and N_S because l_c depends on θ_1.

The final step of the model is to sum the relative contributions of the bending, straightening, and slipping fibre elements. Because all fibre elements are assumed to be initially curved, the energy developed in a single fibre element under deformation is given by the classical equation for the energy developed in an initially-curved bar under deformation,

$$E_b = \frac{Bl_1}{2}(\kappa_1 - \kappa_2)^2, \tag{A.17}$$

where

$$B = E_f\frac{\pi D^4}{64} \tag{A.18}$$

is the bending rigidity of the fibre, E_f is the Young's modulus of elasticity of the fibre, and D is the fibre diameter. This equation applies equally to all three element types. In particular, for slipping elements we simply set κ_2 to zero, to give the energy developed in the element due to its complete straightening prior to slipping. No energy contribution is accounted for after slipping, because work against friction is ignored. Therefore,

$$E_B = E_T = E_b \tag{A.19}$$

$$E_S = \frac{Bl_1}{2}\kappa_1^2 \tag{A.20}$$

The total energy is therefore

$$E_{\text{tot}} = N\left[\int_0^{\theta_c} E_B\Omega(\theta)\sin\theta d\theta_1 + \int_{\theta_c}^{\pi/2}\int_0^{l_c} E_T f(l)\Omega(\theta)\sin\theta dl_1 d\theta_1 + \int_{\theta_c}^{\pi/2}\int_{l_c}^{\infty} E_S f(l)\Omega(\theta)\sin\theta dl_1 d\theta_1\right] \tag{A.21}$$

To complete the model, it is necessary to specify the fibre element length distribution $f(l)$ and the orientation density distribution $\Omega(\theta)\sin\theta$. Lee and Carnaby define these as [72]

$$f(l) = \frac{[(n+1)/\bar{l}]^{n+1}}{n!} l^n e^{-[(n+1)/\bar{l}]l} \tag{A.22}$$

and

$$\Omega(\theta)\sin\theta = \frac{c^2}{(\sqrt{1+c^2\tan^2\theta}\cos\theta)^3}\sin\theta, \tag{A.23}$$

where

$$c = \frac{1+\epsilon_a}{1-\nu_a\epsilon_a}. \tag{A.24}$$

A.3 Analysis

The Lee Carnaby model was re-implemented in Python 2.7 [82], using the IPython interactive computing environment [83]. The symbolic mathematics library SymPy [84] was used to construct the equations, and numerical integration was performed using the NumPy [85] and SciPy [86] libraries.

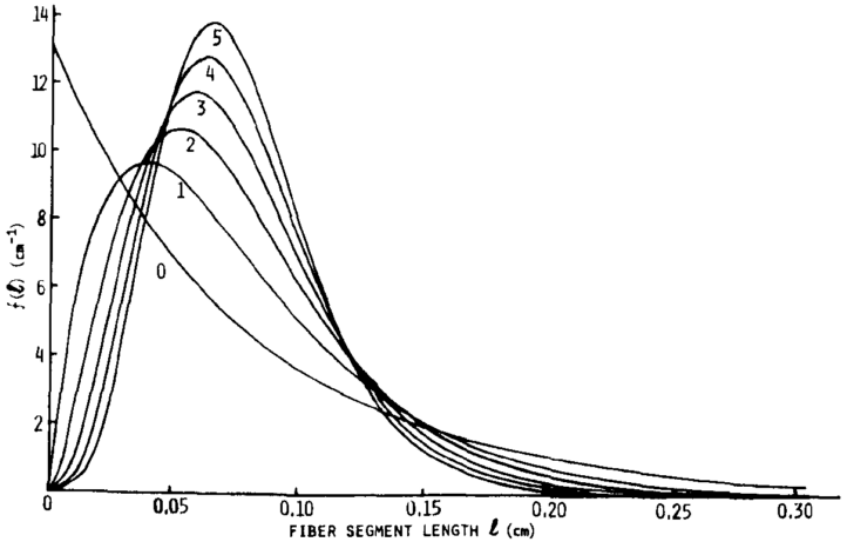
A.3.1 Verification of the re-implementation

Figures A.3 to A.6 present several of the original figures from Lee, Carnaby and Tandon's evaluation paper [72]. Alongside the increase in resolution that comes with modern computing power, there are no discernable discrepancies between any of the figure pairs. This proves the correctness of the re-implemented model (relative to its original FORTRAN implementation).

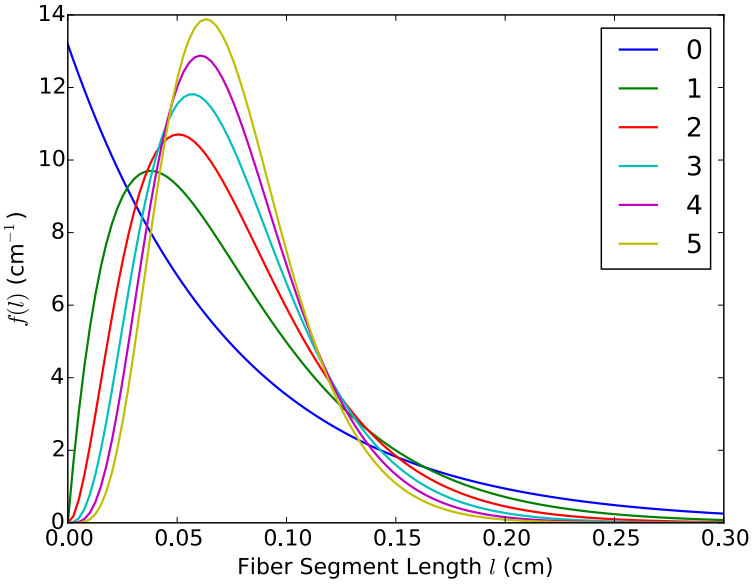
As stated in the previous section, the Lee Carnaby model was designed to have two variables: the compressional strain ϵ_a and the Poisson's ratio ν_a. Figures A.4a and A.5a show how Lee, Carnaby and Tandon calculated energy vs. ν_a curves for specific values of ϵ_a; by the minimum energy principle they then defined the "true" ν_a at each ϵ_a as being that which gave the lowest energy. Figure A.6a is the result of this minimisation process, and its agreement with fig. A.6b indicates that the minimisation routine used in the re-implementation is equivalent to the method used by Lee, Carnaby and Tandon [72].

It is at this point that the added computing power reaps its rewards. Figure A.7 shows the full surface and contour plots corresponding to the Lee Carnaby figures; the modern re-implementation makes these representation feasible to calculate. The origins of the trends shown in figs. A.4 to A.6 are clearly visible:

- figs. A.4 and A.5 are slices through fig. A.7a at $\epsilon_a = -10^{-4}$ and -10^{-2}.
- fig. A.6 is the plot of energy vs. ϵ_a along the (black dotted) ν_a minimisation line in fig. A.7b.

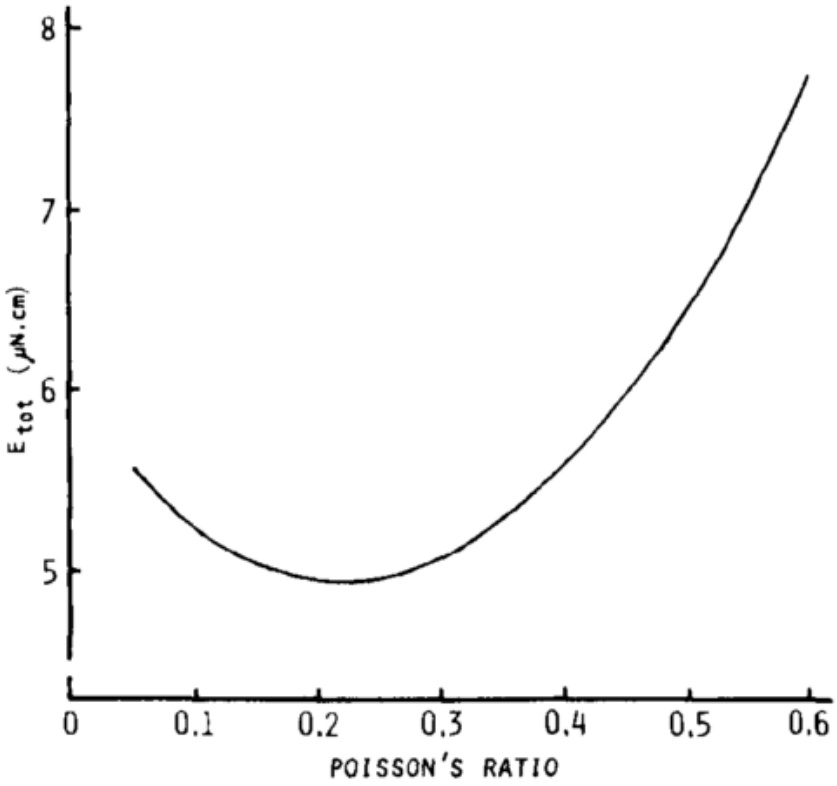


(a) Original [72]

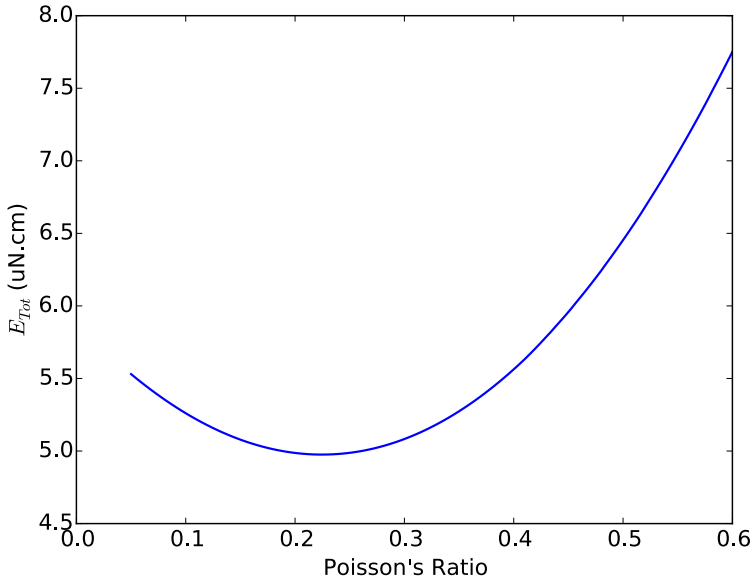


(b) Modern

Figure A.3: Recreation of Lee Carnaby Figure 2.

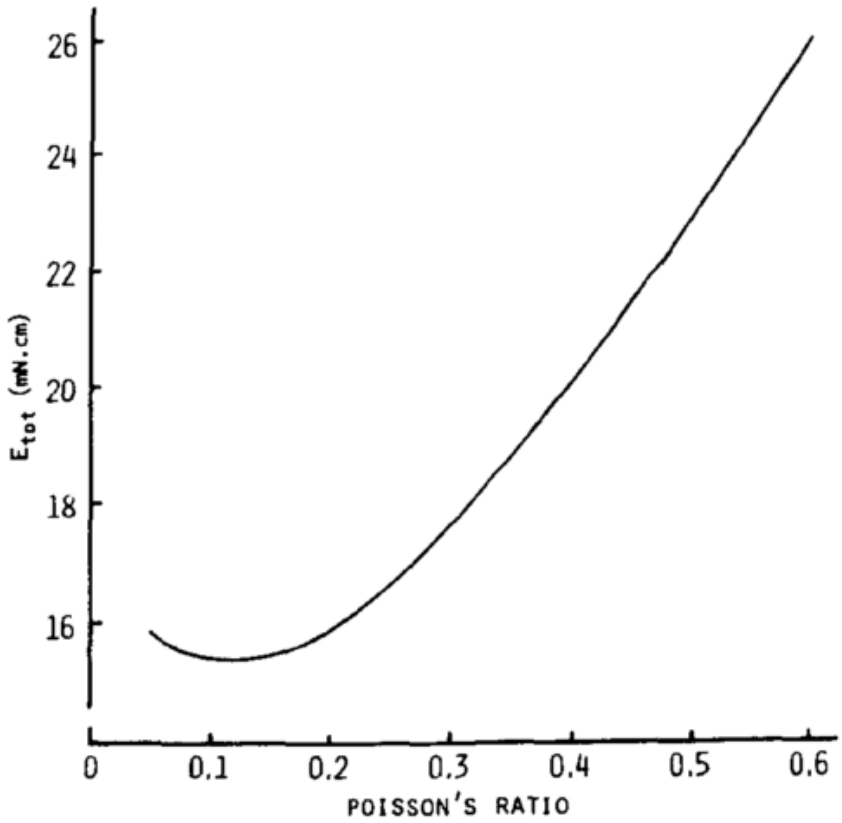


(a) Original [72]

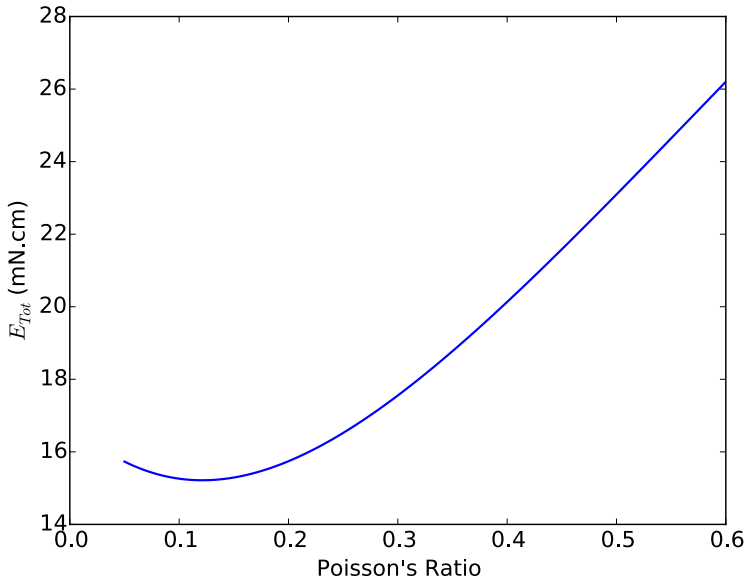


(b) Modern

Figure A.4: Recreation of Lee Carnaby Figure 4.

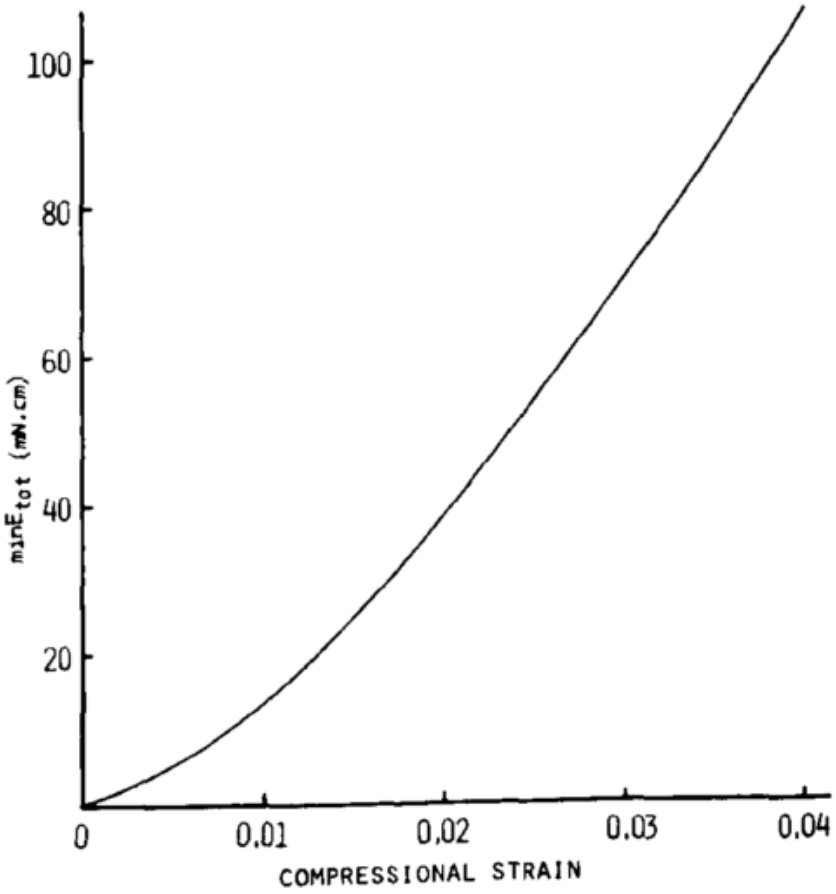


(a) Original [72]

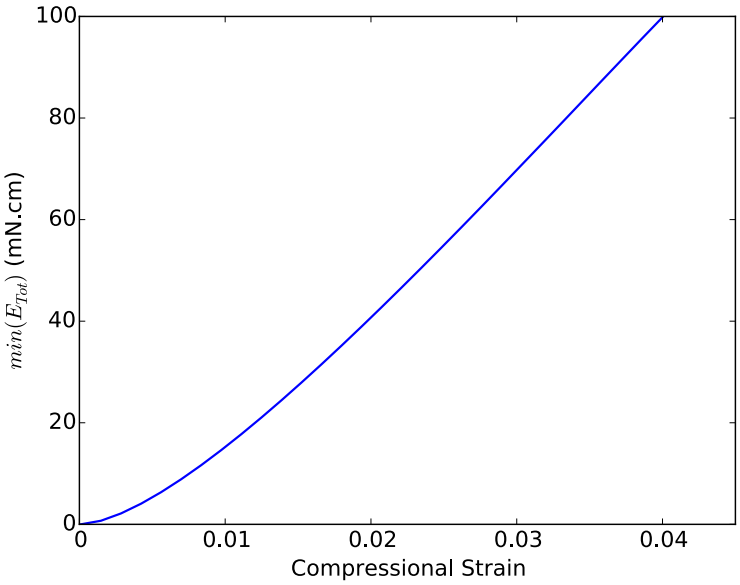


(b) Modern

Figure A.5: Recreation of Lee Carnaby Figure 5.

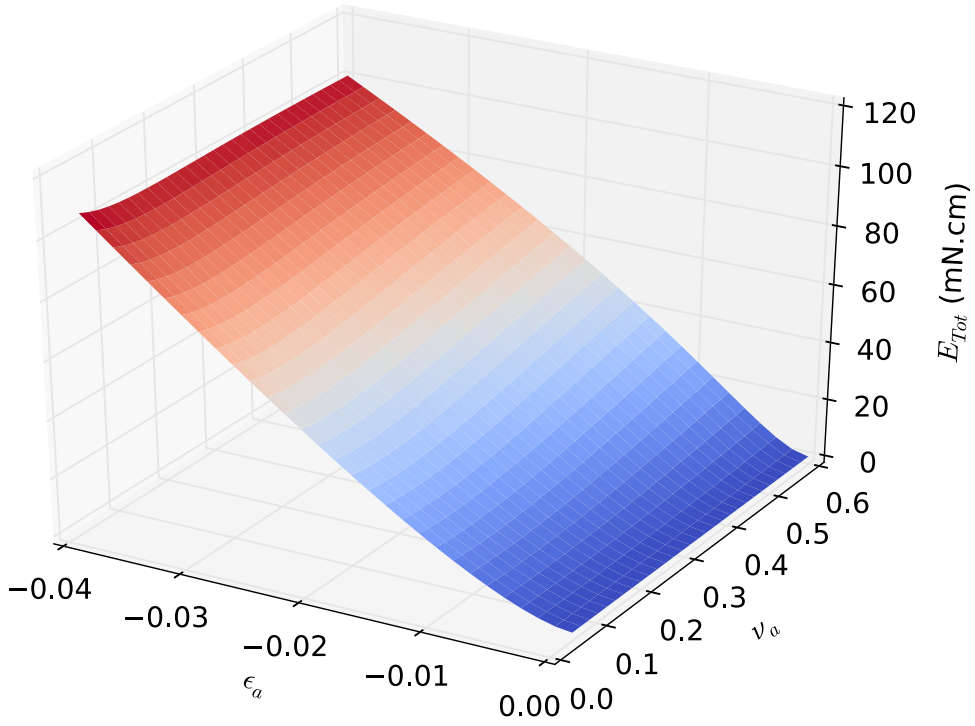


(a) Original [72]

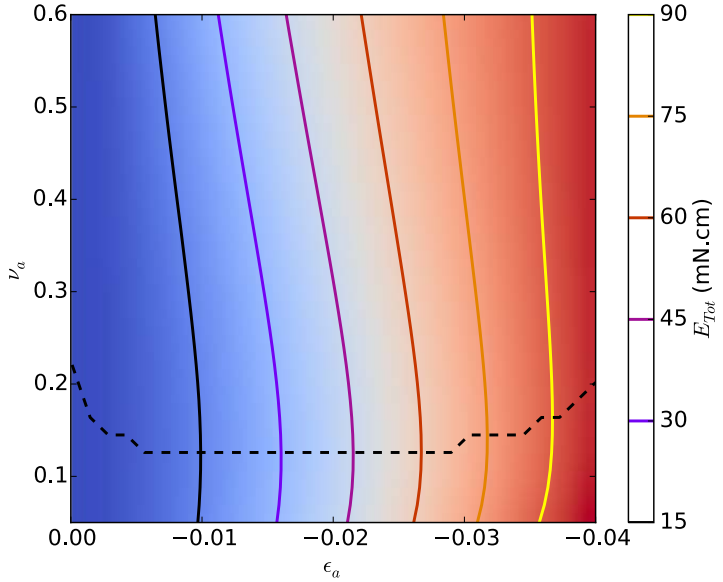


(b) Modern

Figure A.6: Recreation of Lee Carnaby Figure 6.



(a) Surface plot



(b) Contour plot. The dotted line shows the minimum ν_a along the ϵ_a axis.

Figure A.7: Surface and contour plots corresponding to the Lee Carnaby figures.

A.3.2 Issues

The first issue that we note with the Lee Carnaby model is that in the evaluation paper [72], the parameter table states that values of ϵ_a up to -0.1 (ie. 10% compression) were considered, but fig. A.6a was only plotted up to $\epsilon_a = -0.04$.

Figure A.8 is the same plot as fig. A.6a, but plotted up to $\epsilon_a = -0.1$. It is immediately obvious that there is a problem with the energy values being calculated—there is a discontinuity in the energy vs. strain curve at $\epsilon_a = -0.04$.

The cause of this discontinuity is apparent when examining the corresponding surface and contour plots in fig. A.11: the slope of the energy gradient changes beyond $\epsilon_a = -0.04$, and in finding the minimum energy, the ν_a value maxes out at the top of the range of computation. This can be seen in the minimum energy path plotted on fig. A.11b.

Extending the upper bound on ν_a to unrealistic values (fig. A.10) does not resolve the issue; in fact the discontinuity is more pronounced. Clearly, the Lee Carnaby model is fundamentally incompatible with strains beyond 4%.

Figure A.11 shows the energy surface across the entire compression strain space. The cause of the discontinuity is now clear: the energy surface plateaus at high ν_a, around 100 mN cm, while at small ν_a the energy peaks over 750 mN cm; this invalidates the minimisation methodology.

Figure A.12 plots the individual energy components that make up eq. (A.21). The plateau is entirely due to the E_S component, while the energy peak is caused by the E_B component. Based on these graphs, we posit that the failure of the Lee Carnaby model arises from the fact that there is no restriction on slippage. Recall that eq. (A.8) defined a critical length l_c that forms the boundary between straightening fibre segments, and slipping ones. Slipping fibre segments are defined as those which have fully straightened, and therefore must slip (since stretching is a much higher-energy process). We believe that problems arise because friction at the slipping point is ignored. This means that once a fibre segment becomes slipping, it contributes no further energy to the overall strain. Therefore, as ϵ_a increases, the system tends towards all fibres becoming slipping, because that is the lowest energy state.

Figure A.13 confirms our hypothesis. The fraction of bending fibres (fig. A.13a) is highest when ν_a is close to zero, as expected—increasing ν_a spreads the assembly laterally, causing fibres that were bending to straighten. However, very few fibres are ever in the straightening state: fig. A.13b shows that N_T drops very quickly, and fig. A.13c shows that the slipping state is the most preferred state for most values of ϵ_a and ν_a. We can also see the origin of the behaviour transition in fig. A.13b; the contour plot has a bifurcation at $\epsilon_a \approx -0.04$.

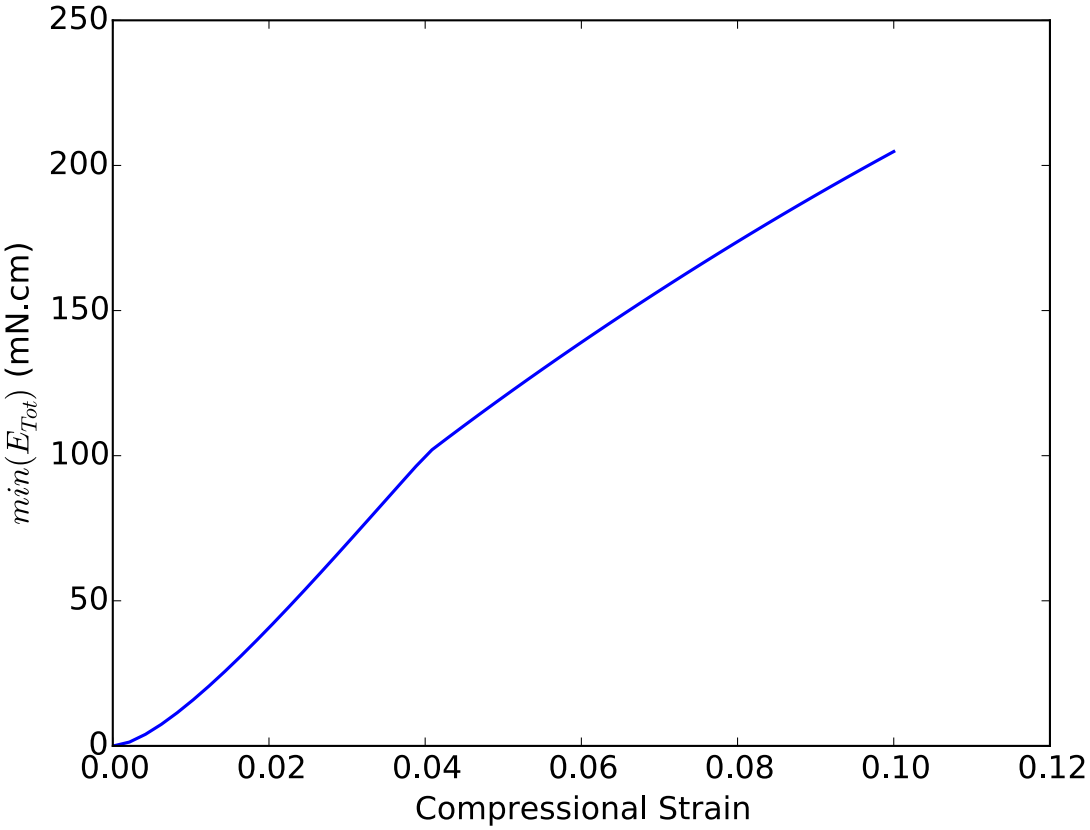
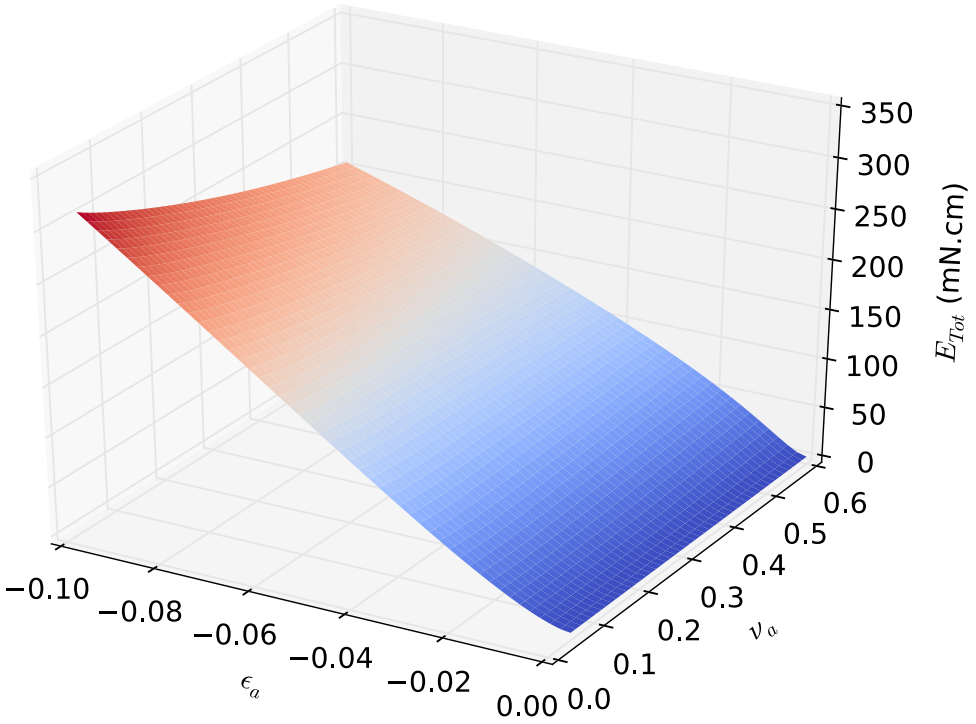
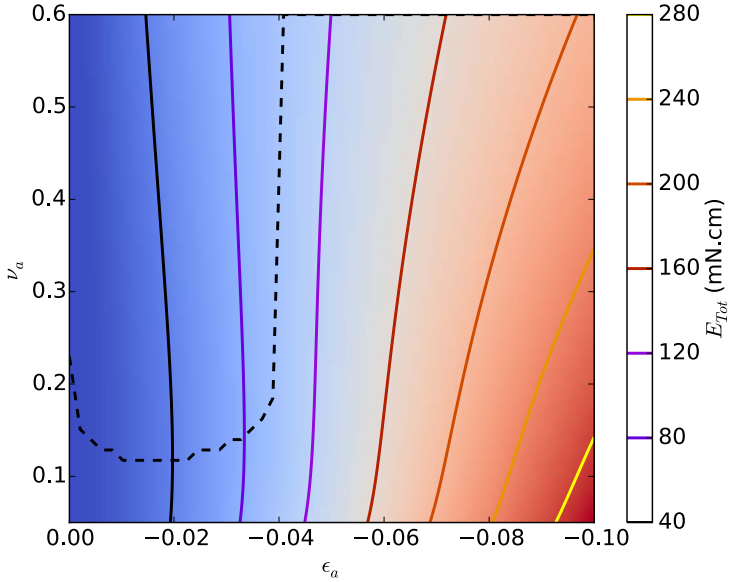


Figure A.8: Energy vs. Compressional strain

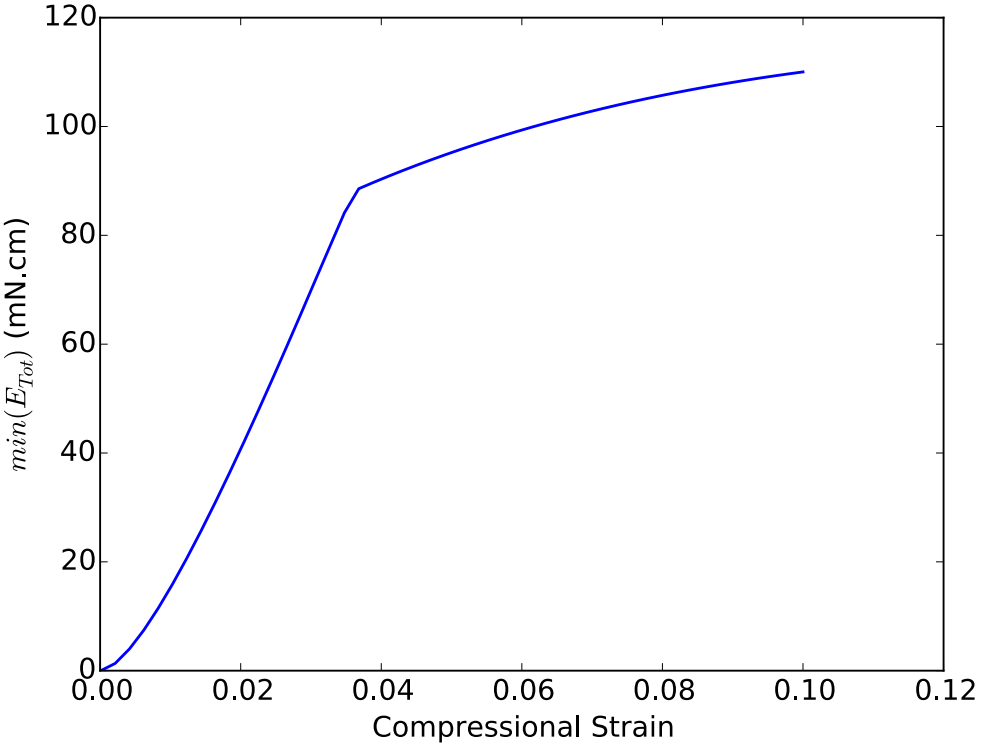


(a) Surface plot

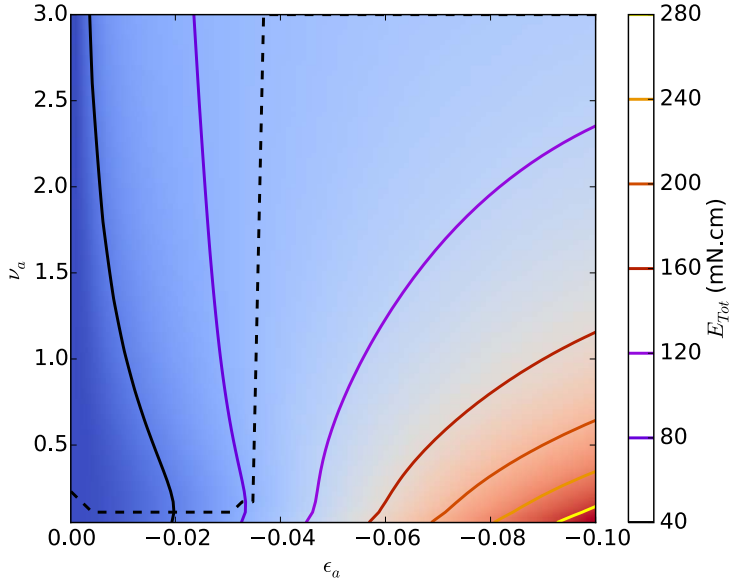


(b) Contour plot. The dotted line shows the minimum ν_a along the ϵ_a axis.

Figure A.9: Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -0.1$.

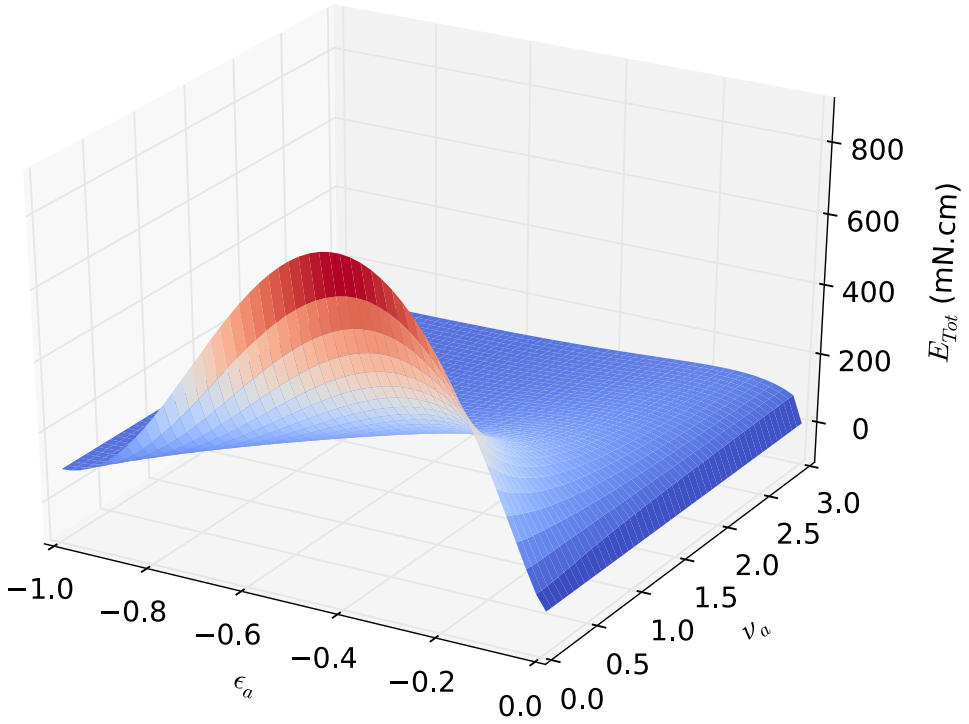


(a) Energy vs. Compressional strain

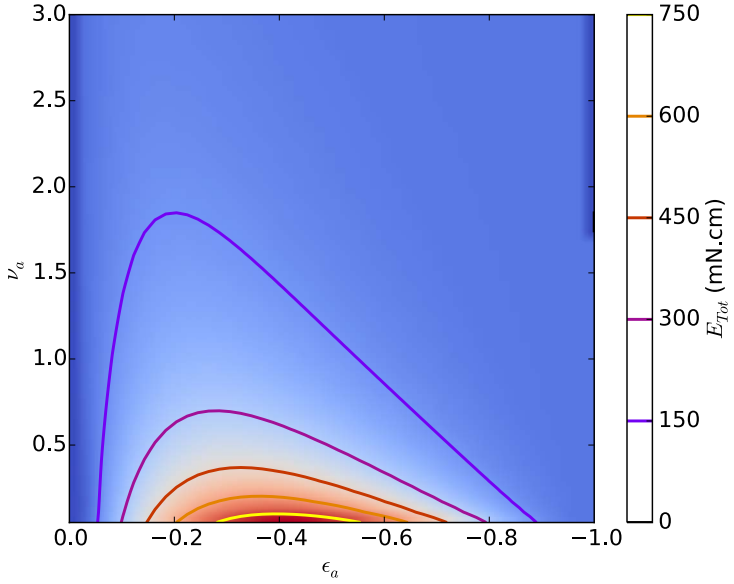


(b) Contour plot. The dotted line shows the minimum ν_a along the ϵ_a axis.

Figure A.10: Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -0.1$ and $\max(\nu_a) = 3$.

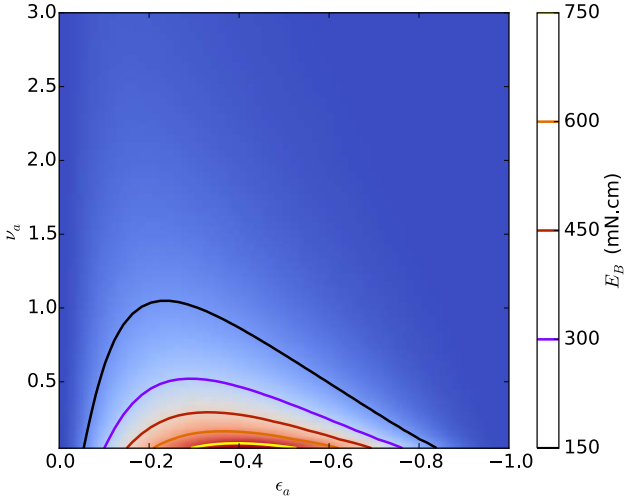


(a) Surface plot

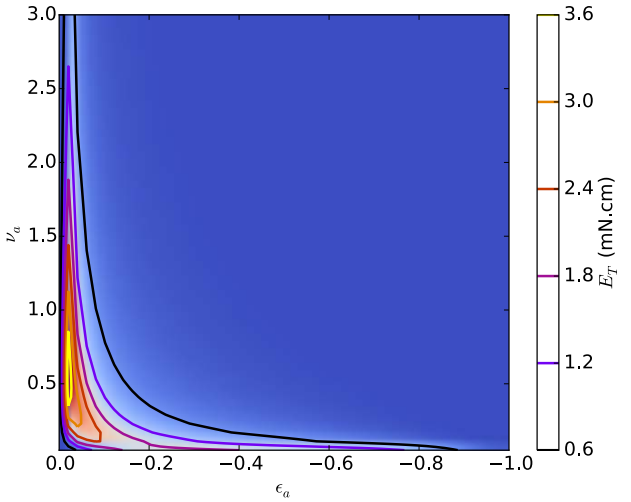


(b) Contour plot. The dotted line shows the minimum ν_a along the ϵ_a axis.

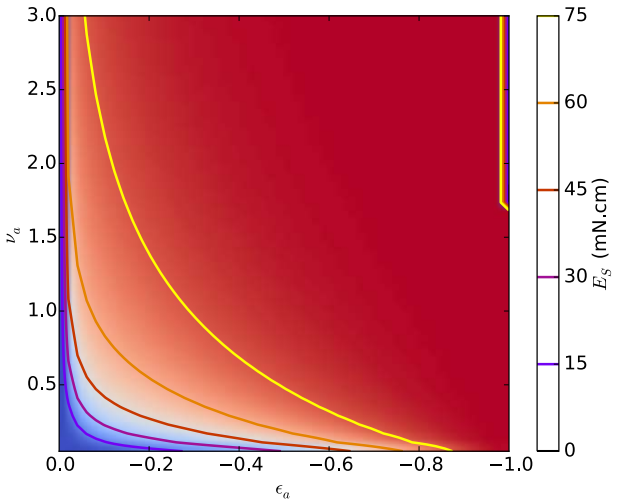
Figure A.11: Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -1$ and $\max(\nu_a) = 3$.



(a) Surface plot of the E_B component

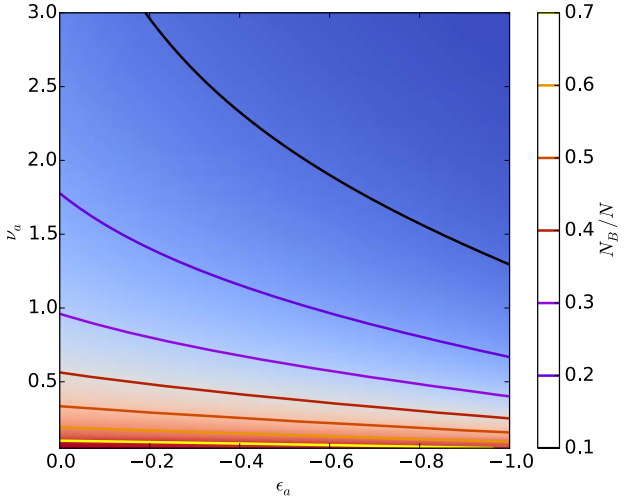


(b) Surface plot of the E_T component

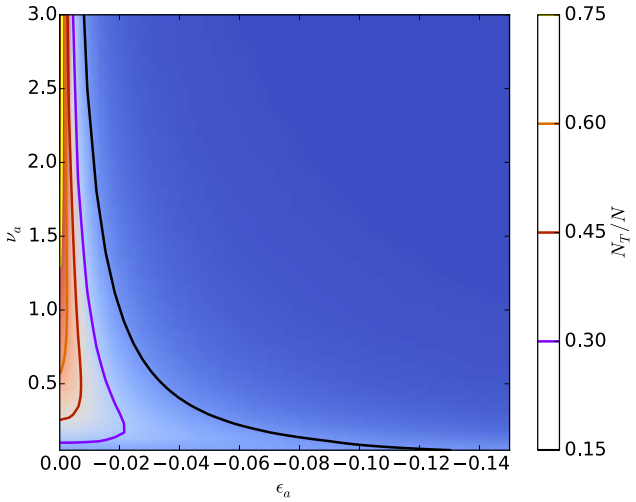


(c) Surface plot of the E_S component

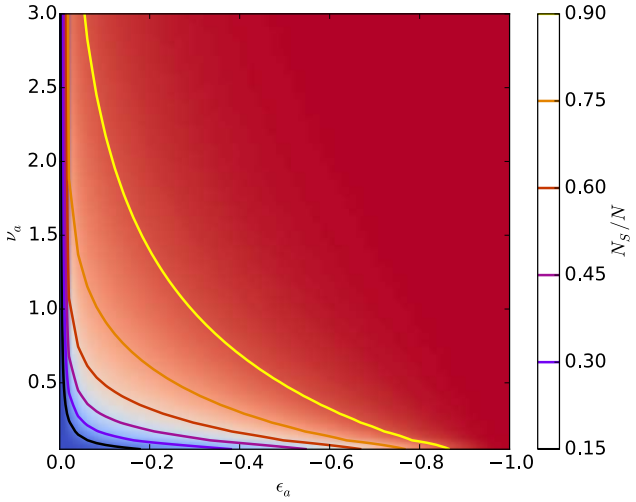
Figure A.12: Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -1$ and $\max(\nu_a) = 3$.



(a) Surface plot of N_B/N



(b) Surface plot of N_T/N



(c) Surface plot of N_S/N

Figure A.13: Energy vs. strain plot, and surface and contour plots, for $\max(\epsilon_a) = -1$ and $\max(\nu_a) = 3$.

Appendix B

Code listings

B.1 SimpleSphere.ipynb

In this notebook, we implement the simple knoppy web model from section 4.2. As described there, we construct the model equations to have the following variables:

- c, the distance by which the sphere has been compressed.
- r', the measure of the sphere's spread.

We integrate over these variables to obtain the energy map of the problem domain. Then we minimise over the total energy U_T to obtain the relationship between unit cell compression and component compression.

We start with imports, as usual.

```
In [ ]: local  = True
        run    = True
        titles = False
```

```
In [ ]: from ipyparallel import Client
        if local:
            rc = Client()
        else:
            rc = Client('path/to/ipcontroller-client.json')
        dview = rc[:]
        dview.clear()

        if local:
            # Only run if the engines are on the same machine.
            # On a different machine, make sure to copy equations/
            # and utils.py into the cwd of the engines.
            model_dir = %pwd
            dview.push({'model_dir': model_dir})
            %px import os
            %px os.chdir(model_dir)
```

```
In [ ]: from __future__ import division
        with dview.sync_imports():
            import numpy as np
            from scipy.integrate import quad, dblquad
```

```
            from sympy import lambdify, symbols

            from equations import simple_sphere as ss
            from utils import lru_cache

        %px np = numpy
        %px ss = simple_sphere

        from ipywidgets import interact
        import math
        from time import time

        import plot as p
```

Constants

We define the following constants from the model:

```
In [ ]: r0 = 0.25
        mu = 0.5
        h = 0.005
        Ek = ((0.05 * 10**8)/9.8) # 0.05 GPa in gf/cm^2
```

```
In [ ]: dview.push(dict(r0=r0, mu=mu, h=h, Ek=Ek));
```

```
In [ ]: %%px --local
        subs_vars = dict(zip([ss.r0, ss.mu, ss.h, ss.Ek], [r0, mu, h, Ek]))
```

Integration

Now we define the integration itself.

```
In [ ]: %%px --local
        class Numeric(object):
            def __init__(self, subs_vars):
                self.UMF_int_inner = lambdify((ss.theta, ss.c, ss.rd),
                                              ss.UMF_int_inner.subs(subs_vars))
                self.UMS_int = lambdify((ss.r, ss.theta, ss.c, ss.rd),
                                        ss.UMS_int.subs(subs_vars))

                self.r_llim = ss.r_llim.subs(subs_vars)
                self.r_ulim = ss.r_ulim.subs(subs_vars)
                self.theta_c = lru_cache()(lambdify(
                        (ss.c), ss.theta_c.subs(subs_vars)))
                self.theta_ulim = ss.theta_ulim.subs(subs_vars)

                self.UMF_fac_inner = np.vectorize(lambdify(
                        (ss.c, ss.rd), ss.UMF_fac_inner.subs(subs_vars)))
                self.UMF_second_part = np.vectorize(lambdify(
                        (ss.c, ss.rd), ss.UMF_second_part.subs(subs_vars)))
                self.UMF_fac_outer = lambdify((symbols('y')),
```

```
                                                ss.UMF_fac_outer.subs(subs_vars))(0)
            self.UMS_fac = lambdify((symbols('y')),
                                    ss.UMS_fac.subs(subs_vars))(0)

            self.UB = lru_cache()(lambdify((ss.c), ss.UB.subs(subs_vars)))

            self.vec_UMFint = np.vectorize(self.UMFint)
            self.vec_UMSint = np.vectorize(self.UMSint)
            self.vec_UB = np.vectorize(self.UB)

        def UMFint(self, c, rd):
            return quad(self.UMF_int_inner, 0, self.theta_c(c), (c, rd))[0]

        def UMSint(self, c, rd):
            return dblquad(self.UMS_int,
                           self.theta_c(c), self.theta_ulim,
                           lambda x: self.r_llim, lambda x: self.r_ulim,
                           (c, rd))[0]

        def integrate(self, c, rd):
            UMF = self.UMF_fac_outer * (self.UMF_fac_inner(c, rd) *
                                        self.vec_UMFint(c, rd) +
                                        self.UMF_second_part(c, rd))
            UMS = self.UMS_fac * self.vec_UMSint(c, rd)
            UB = self.vec_UB(c)
            return UMF, UMS, UB

    def evaluate():
        return Numeric(subs_vars).integrate(c, rd)
```

Then we define the limits of integration.

In [ ]:
```
cSize = 360
rdSize = 360

clim = (0.001, r0-h)
rdlim = (0.001, 2*r0)
cVals = np.linspace(*(clim + (cSize,)))
rdVals = np.linspace(*(rdlim + (rdSize,)))
c, rd = np.meshgrid(cVals, rdVals)
```

Finally, we perform the integration itself, and calculate the total energy U_T.

In [ ]:
```
if run:
    dview.scatter('c', c)
    dview.scatter('rd', rd)

    start = time()
    %px UMF, UMS, UB = evaluate()
    print 'Integration took %d seconds' % (time() - start)
```

```
            UMF = dview.gather('UMF').result
            UMS = dview.gather('UMS').result
            UB = dview.gather('UB').result
```

The following two cells are present to allow saving and loading of the data.

```
In [ ]: if run:
            np.savez('numeric-ss-%d-%d-r0%.2f-h%.3f-mu%.1f.npz' %
                     (cSize, rdSize, r0, h, mu),
                     c=c, rd=rd,
                     UMF=UMF, UMS=UMS, UB=UB)
```

```
In [ ]: if not run:
            with np.load('numeric-ss-%d-%d-r0%.2f-h%.3f-mu%.1f.npz' %
                         (cSize, rdSize, r0, h, mu)) as a:
                c = a['c']
                rd = a['rd']
                UMF = a['UMF']
                UMS = a['UMS']
                UB = a['UB']
```

From the returned energy components, we can calculate the total energy:

```
In [ ]: UT = 2*UMF + UMS + UB
```

Physical behaviour

We want to see how our model behaves, and the most intuitive way to do so is to plot the compression behaviour of the knop interactively.

```
In [ ]: plot_range = [-0.4, 0.4, -0.3, 0.3]
```

```
        def interact_sphere(level):
            j = level - 1
            cVal = cVals[j]
            rdMin = 0
            UTmin = np.amax(UT)
            UMFmin = np.amax(UMF)
            UMSmin = np.amax(UMS)
            UBmin = np.amax(UB)
            for i in np.arange(rdSize):
                if UT[i,j] < UTmin:
                    UTmin = UT[i,j]
                    UMFmin = UMF[i,j]
                    UMSmin = UMS[i,j]
                    UBmin = UB[i,j]
                    rdMin = rdVals[i]

            rdd = float(ss.rdd.subs({ss.r0: r0, ss.c: cVal}))
```

```
            p.plot_simple_sphere(rdMin, rdd, h, plot_range, (UMFmin, UMSmin, UBmin))
            print 'c:      %f cm' % cVal
            print 'UMF:    %f gf.cm' % UMFmin
            print 'UMS:    %f gf.cm' % UMSmin
            print 'UB:     %f gf.cm' % UBmin
            print "r0/r'': %f" % (r0/rdd)
```

```
In [ ]: %matplotlib inline
        interact(interact_sphere, level=(1,cSize))
```

Analysis

We now plot the figures used in the thesis.

We plot U_T as a surface plot to show the overall landscape, and a contour plot showing both the energy field and the minimum energy path (fig. 4.5).

```
In [ ]: p.plotSurface(c, rd, UT/1000,
                      titles and r'$U_T$' or None,
                      r'$c$ (cm)', r"$r'$ (cm)", r'$U_T$ (kgf.cm)',
                      logNorm=True)

        rdzero = c.min(0)
        rdone = c.min(0)*(math.pi/2)
        p.plotContour(c, rd, UT/1000, clim, rdlim,
                      titles and r'$U_T$' or None,
                      r'$c$ (cm)', r"$r'$ (cm)",
                      logNorm=True, showMin=True,
                      zlabel=r'$U_T$ (kgf.cm)',
                      extraLine=[(c.min(0), rdzero),(c.min(0), rdone)])
```

Now we break it down by component.

First, U_{MF} (fig. 4.7):

```
In [ ]: p.plotSurface(c, rd, UMF/1000,
                      titles and r'$U_{M_F}$' or None,
                      r'$c$ (cm)', r"$r'$ (cm)", r'$U_{M_F}$ (kgf.cm)',
                      logNorm=True)

        rdzero = c.min(0)*(math.pi/2)
        p.plotContour(c, rd, UMF/1000, clim, rdlim,
                      titles and r'$U_{M_F}$' or None,
                      r'$c$ (cm)', r"$r'$ (cm)",
                      logNorm=True, showMin=True,
                      zlabel=r'$U_{M_F}$ (kgf.cm)',
                      extraLine=(c.min(0), rdzero))
```

Next, U_{MS} (fig. 4.8):

```
In [ ]: p.plotSurface(c, rd, UMS/1000,
                      titles and r'$U_{M_S}$' or None,
                      r'$c$ (cm)', r"$r'$ (cm)", r'$U_{M_S}$ (kgf.cm)',
```

```
                    logNorm=True)

        rdzero = c.min(0)
        thetac = np.vectorize((lambdify((ss.c), ss.theta_c.subs(subs_vars))))(c.min(0))
        rdone = r0*np.vectorize(math.sin)(thetac)
        p.plotContour(c, rd, UMS/1000, clim, rdlim,
                    titles and r'$U_{M_S}$' or None,
                    r'$c$ (cm)', r"$r'$ (cm)",
                    logNorm=True, showMin=True,
                    zlabel=r'$U_{M_S}$ (kgf.cm)',
                    extraLine=[(c.min(0), rdzero),(c.min(0), rdone)])
```

And finally U_B (fig. 4.9):

```
In [ ]: p.plotSurface(c, rd, UB,
                    titles and r'$U_B$' or None,
                    r'$c$ (cm)', r"$r'$ (cm)", r'$U_B$ (gf.cm)',
                    logNorm=True)

        p.plotContour(c, rd, UB, clim, rdlim,
                    titles and r'$U_B$' or None,
                    r'$c$ (cm)', r"$r'$ (cm)",
                    zlabel=r'$U_B$ (gf.cm)',
                    logNorm=True)
```

We can also show the evolution of the various energy terms as the sphere is compressed along the minimum total energy path. At each value of c we find the value of r' for which U_T is a minimum, and then record the corresponding values of U_{M_F}, U_{M_S} and U_B.

```
In [ ]: cmin = np.zeros(cSize)
        UTmin = np.empty(cSize)
        UMFmin = np.empty(cSize)
        UMSmin = np.empty(cSize)
        UBmin = np.empty(cSize)

        UTmin[:] = np.amax(UT)
        UMFmin[:] = np.amax(UMF)
        UMSmin[:] = np.amax(UMS)
        UBmin[:] = np.amax(UB)

        for i in np.arange(rdSize):
            for j in np.arange(cSize):
                if UT[i,j] < UTmin[j]:
                    cmin[j] = c[i,j]
                    UTmin[j] = UT[i,j]
                    UMFmin[j] = UMF[i,j]
                    UMSmin[j] = UMS[i,j]
                    UBmin[j] = UB[i,j]
```

We plot U_T against c (fig. 4.6):

```
In [ ]: p.plotGraph(cmin, UTmin,
                    titles and r'$min(U_T)$' or None,
                    r'$c$ (cm)', r'$U_T$ (gf.cm)',
                    linewidth=2)
```

And the corresponding values of the energy components (fig. 4.10):

```
In [ ]: p.plotGraph(cmin, [2*UMFmin, UMSmin, UBmin],
                    titles and 'Minimised energy terms' or None,
                    r'$c$ (cm)', 'Energy (gf.cm)',
                    linewidth=2,
                    logY=False) # Change to True for figure (b)
```

Other views

The following views were not used in the thesis, but were useful for checking certain areas of the model.

```
In [ ]: %matplotlib inline
        from matplotlib import gridspec
        import matplotlib.pyplot as plt
        import figures

        def interact_theta_f(cNum, rdNum, qtheta):
            cVal = cVals[cNum]
            rdVal = rdVals[rdNum]
            subs = dict(zip([ss.r0, ss.h, ss.c, ss.rd], [r0, h, cVal, rdVal]))

            rdd = figures.get(ss.rdd, subs)
            theta_c = figures.get(ss.theta_c, subs)
            theta = theta_c + (ss.theta_ulim-theta_c)*qtheta
            r = np.linspace(r0-h, r0+h, 3)
            P = []
            for point in r:
                P.append({ss.r: point, ss.theta: theta})

            fig = plt.figure()
            gs = gridspec.GridSpec(1, 3, width_ratios=[1,1,2])
            # Uncompressed
            ax = figures.init_axis(gs[0], [0, r0+h+0.1, -r0-h-0.1, r0+h+0.1])
            figures.half_uncompressed_sphere_outer(ax, subs, P)
            # Compressed
            ax = figures.init_axis(gs[1], [0, rdVal+rdd+h+0.1, -r0-h-0.1, r0+h+0.1])
            figures.half_compressed_sphere_outer(ax, subs, P)

            plt.subplot(gs[2])
            ax = plt.gca()
            rdCol = rd[:,cNum]
            ephi_old_func = np.vectorize(lambdify(
                    (ss.rd, ss.r, ss.theta),
                    ss.UMS_ephi_old.subs(dict(zip(
```

```
                                [ss.r0, ss.h, ss.c], [r0, h, cVal])))))
        ephi_func = np.vectorize(lambdify(
                (ss.rd, ss.r, ss.theta),
                ss.UMS_ephi.subs(dict(zip([ss.r0, ss.h, ss.c], [r0, h, cVal])))))
        for point in r:
            ax.plot(rdCol, ephi_old_func(rdCol, point, theta))
        for point in r:
            ax.plot([rdVal], [ephi_old_func(rdVal, point, theta)], 'or')
        for point in r:
            ax.plot(rdCol, ephi_func(rdCol, point, theta))
        for point in r:
            ax.plot([rdVal], [ephi_func(rdVal, point, theta)], 'or')

        # Remove outer whitespace
        gs.tight_layout(fig, pad=0)
        plt.show()

        ephi_func = np.vectorize(lambdify((ss.theta, ss.r),
                                          ss.UMS_ephi.subs(subs)))
        ephi = ephi_func(theta, r)
        print 'ephi:'
        for line in ephi:
            print line

    interact(interact_theta_f,
             cNum=(0, cSize-1), rdNum=(0, rdSize-1), qtheta=(0, 1, 0.001))
```

B.2 SimpleSphereParams.py

In this notebook, we show the effect of the various model parameters by plotting the sphere energy against compression for several values of each parameter.

```
In [ ]: from __future__ import division
        import matplotlib.pyplot as plt
        import numpy as np
        import plot as p

        cSize = 360
        rdSize = 360

        params = [
            ('r0%.2f-h0.005-mu0.5', [0.1, 0.25, 0.5]),
            ('r00.25-h%.3f-mu0.5', [0.005, 0.01, 0.015]),
            ('r00.25-h0.005-mu%.1f', [0.1, 0.3, 0.5]),
        ]
```

We load the datafiles generated by `SimpleSphere.ipynb` (appendix B.1) and calculate the minimum energy path:

```
In [ ]: # TWEAK-START Change for h and mu
        param = 0
        # TWEAK-END

        cmins = []
        UTs = []
        UMFs = []
        UMSs = []
        UBs = []

        for pval in params[param][1]:
            with np.load('numeric-ss-%d-%d-%s.npz' %
                         (cSize, rdSize, params[param][0] % pval)) as a:
                c = a['c']
                rd = a['rd']
                UMF = a['UMF']
                UMS = a['UMS']
                UB = a['UB']

            UT = 2*UMF + UMS + UB

            cmin = np.zeros(cSize)
            UTmin = np.empty(cSize)
            UMFmin = np.empty(cSize)
            UMSmin = np.empty(cSize)
            UBmin = np.empty(cSize)

            UTmin[:] = np.amax(UT)
            UMFmin[:] = np.amax(UMF)
            UMSmin[:] = np.amax(UMS)
            UBmin[:] = np.amax(UB)

            for i in np.arange(rdSize):
                for j in np.arange(cSize):
                    if UT[i,j] < UTmin[j]:
                        cmin[j] = c[i,j]
                        UTmin[j] = UT[i,j]
                        UMFmin[j] = UMF[i,j]
                        UMSmin[j] = UMS[i,j]
                        UBmin[j] = UB[i,j]

            cmins.append(cmin)
            UTs.append(UTmin)
            UMFs.append(UMFmin)
            UMSs.append(UMSmin)
            UBs.append(UBmin)
```

And then we plot U_T vs c along with the various energy components (figs. 4.11 to 4.13):

```
In [ ]: def plotVals(vals, ylabel):
```

```
        fig = plt.figure()
        ax = fig.add_subplot(111)
        for i in range(0, len(params[param][1])):
            ax.plot(cmins[i], vals[i], linewidth=2)

        ax.set_xlabel(r'$c$ (cm)')
        ax.set_ylabel(ylabel)

        plt.show()

    plotVals(UTs, r'$U_T$ (gf.cm)')
    plotVals(UMFs, r'$U_{M_F}$ (gf.cm)')
    plotVals(UMSs, r'$U_{M_S}$ (gf.cm)')
    plotVals(UBs, r'$U_B$ (gf.cm)')
```

B.3 BasicKnoppyWebUnitCell.ipynb

In this notebook, we implement the basic knoppy web unit cell from section 4.3.1. As described there, we construct the model equations to have the following variables:

- d, the distance by which the unit cell has been compressed.
- q_c, the ratio between sphere compression and web compression.
- r', the measure of the sphere's spread.

We integrate over these variables to obtain the energy map of the problem domain. Then we minimise over r' and the total energy U_{Tot} to finally obtain the relationship between unit cell compression and component compression.

In [ ]:
```
local  = True
run    = True
titles = False
```

In [ ]:
```
from ipyparallel import Client
if local:
    rc = Client()
else:
    rc = Client('path/to/ipcontroller-client.json')
dview = rc[:]
dview.clear()

if local:
    # Only run if the engines are on the same machine.
    # On a different machine, make sure to copy equations/
    # and utils.py into the cwd of the engines.
    model_dir = %pwd
    dview.push({'model_dir': model_dir})
    %px import os
    %px os.chdir(model_dir)
```

In [ ]:
```
from __future__ import division
with dview.sync_imports():
```

```
            from math import pi
            import numpy as np
            from scipy.integrate import quad, dblquad
            from sympy import lambdify, symbols

            from equations import simple_sphere as ss, basic_kw as kw, vanwyk as vw
            from utils import lru_cache

        %px np = numpy
        %px ss = simple_sphere
        %px kw = basic_kw
        %px vw = vanwyk

        from ipywidgets import interact
        from time import time

        import plot as p
```

Constants

We start by defining constants for the model.

Web-specific

The general van Wyk equation has a few constants that we define here.

```
In [ ]: Ef = 3.98 * 10**7 # gf/cm^2 == 3.1 N/tex
        rho_f = 1.3 # g/cm^3

        # TWEAK-START Uncomment as appropriate
        #K = 0.001
        K = [0.001, 0.01, 0.1]
        rho_w = 0.03 # g/cm^3
        #rho_w = [0.01, 0.05, 0.1] # g/cm^3
        # TWEAK-END
```

Sphere-specific

The constants below are the same as we used in our analysis of the simple sphere model.

```
In [ ]: r0 = 0.25 # cm
        mu = 0.5
        h = 0.005 # cm
        Ek = ((0.05 * 10**8)/9.8) # gf/cm^2 == 0.05 GPa
```

General constants

Finally, we have constants specific to the knoppy web unit cell. The only one is t_0, the initial thickness of the web section.

```
In [ ]: t0 = 0.25 # cm
```

```
In [ ]: dview.push(dict(K=K, Ef=Ef, rho_w=rho_w, rho_f=rho_f))
        dview.push(dict(r0=r0, mu=mu, h=h, Ek=Ek))
        dview.push(dict(t0=t0))
```

```
In [ ]: %%px --local
        def create_subs_vars(i):
            ss_subs_vars = dict(zip(
                    [ss.r0, ss.mu, ss.h, ss.Ek],
                    [   r0,    mu,    h,    Ek]))
            # TWEAK-START Uncomment as appropriate
            lee_subs_vars = dict(zip(
                    [vw.K, vw.Ef, vw.rho_w, vw.rho_f],
                    [K[i],    Ef,    rho_w,    rho_f]))
        #     lee_subs_vars = dict(zip(
        #             [vw.K, vw.Ef, vw.rho_w,    vw.rho_f],
        #             [   K,    Ef,    rho_w[i],    rho_f]))
            # TWEAK-END
            kw_subs_vars = dict(zip([kw.t0], [t0]))

            subs_vars = {}
            subs_vars.update(ss_subs_vars)
            subs_vars.update(lee_subs_vars)
            subs_vars.update(kw_subs_vars)
            return subs_vars

        # TWEAK-START Uncomment as appropriate
        increments = len(K)
        #increments = len(rho_w)
        filevarname = 'K'
        #filevarname = 'rhow'
        # TWEAK-END
```

Helper methods

Next we define some helper methods.

The first helper method calculates c as a function of d and q_c.

```
In [ ]: %%px --local
        cmax = (r0-h)*7/8 # cm
        tmax = t0*7/8

        def cVal(qc, d):
            if d <= cmax and d <= tmax:
                return qc*d
            elif d <= cmax and d > tmax:
                return qc*d + (1-qc)*(d-tmax)
            elif d > cmax and d <= tmax:
                return qc*cmax
            else:
                return qc*cmax + (1-qc)*(d-tmax)
```

```
In [ ]: %%px --local
        def rdVal(qrd, c):
            return qrd*(c+t0)
```

Integration

Below, we define the integration itself.

```
In [ ]: %%px --local
        vw_subs = dict(zip([vw.v, vw.v0], [kw.vc, kw.v0]))

        class Numeric(object):
            def __init__(self, subs_vars, cVal, rdVal):
                # c(qc, d)
                self.cVal = lru_cache()(cVal)
                # rd(qrd, c)
                self.rdVal = lru_cache()(rdVal)

                # Simple sphere

                self.UMF_int_inner = lambdify((ss.theta, ss.c, ss.rd),
                                              ss.UMF_int_inner.subs(subs_vars))
                self.UMS_int = lambdify((ss.r, ss.theta, ss.c, ss.rd),
                                        ss.UMS_int.subs(subs_vars))

                self.r_llim = ss.r_llim.subs(subs_vars)
                self.r_ulim = ss.r_ulim.subs(subs_vars)
                self.theta_c = lru_cache()(lambdify((ss.c),
                                                    ss.theta_c.subs(subs_vars)))
                self.theta_ulim = ss.theta_ulim.subs(subs_vars)

                self.UMF_fac_inner = lambdify((ss.c, ss.rd),
                                              ss.UMF_fac_inner.subs(subs_vars))
                self.UMF_second_part = lambdify((ss.c, ss.rd),
                                                ss.UMF_second_part.subs(subs_vars))
                self.UMF_fac_outer = lambdify((symbols('y')),
                                              ss.UMF_fac_outer.subs(subs_vars))(0)
                self.UMS_fac = lambdify((symbols('y')),
                                        ss.UMS_fac.subs(subs_vars))(0)

                self.UB_val = lru_cache()(lambdify((ss.c), ss.UB.subs(subs_vars)))

                # van Wyk

                self.U_val = lambdify((kw.d, ss.c, ss.rd),
                                      vw.U.subs(vw_subs).subs(subs_vars))

                # Vector forms
                self.vec_UMFint = np.vectorize(self.UMFint, otypes='d')
                self.vec_UMSint = np.vectorize(self.UMSint, otypes='d')
```

```
            self.vec_UB     = np.vectorize(self.UB,     otypes='d')
            self.vec_U      = np.vectorize(self.U,      otypes='d')

        def UMFint(self, d, qc, qrd):
            c = self.cVal(qc, d)
            rd = self.rdVal(qrd, c)
            return (self.UMF_fac_inner(c, rd) *
                    quad(self.UMF_int_inner,
                         0, self.theta_c(c),
                         (c, rd))[0]) + self.UMF_second_part(c, rd)

        def UMSint(self, d, qc, qrd):
            c = self.cVal(qc, d)
            rd = self.rdVal(qrd, c)
            return dblquad(self.UMS_int,
                           self.theta_c(c), self.theta_ulim,
                           lambda x: self.r_llim, lambda x: self.r_ulim,
                           (c, rd))[0]

        def UB(self, d, qc):
            c = self.cVal(qc, d)
            return self.UB_val(c)

        def U(self, d, qc, qrd):
            c = self.cVal(qc, d)
            rd = self.rdVal(qrd, c)
            return self.U_val(d, c, rd)

        def integrate(self, d, qc, qrd):
            UMF = self.UMF_fac_outer * self.vec_UMFint(d, qc, qrd)
            UMS = self.UMS_fac * self.vec_UMSint(d, qc, qrd)
            UB = self.vec_UB(d, qc)

            U = self.vec_U(d, qc, qrd)

            return UMF, UMS, UB, U
```

```
In [ ]: %%px --local
        def evaluate():
            return Numeric(create_subs_vars(i), cVal, rdVal).integrate(d, qc, qrd)
```

To perform the integration, we define the limits of integration of the knoppy web unit cell, and generate the mesh grids.

```
In [ ]: dlim = (h, cmax+tmax)
        qclim = (0.001, 0.999)
        qrdlim = (0.001, 0.999)

        dSize = 200
        qcSize = 200
```

```
        qrdSize = 200

        dVals = np.linspace(*(dlim + (dSize,)))
        qcVals = np.linspace(*(qclim + (qcSize,)))
        qrdVals = np.linspace(*(qrdlim + (qrdSize,)))
        d, qc, qrd = np.meshgrid(dVals, qcVals, qrdVals)
```

Now we run the integration. This is a slow step; the simple sphere part doesn't take long, but the web calculations require performing a double integral over a three-dimensional space.

```
In [ ]: def run_simulation():
            dview.scatter('d', d)
            dview.scatter('qc', qc)
            dview.scatter('qrd', qrd)

            start = time()
            last = start
            for i in range(0, increments):
                dview.push(dict(i=i))
                %px UMF, UMS, UB, UW = evaluate()
                now = time()
                print 'Integration %d took %d seconds' % (i+1, now - last)
                last = now

                UMF.append(dview.gather('UMF').result)
                UMS.append(dview.gather('UMS').result)
                UB.append(dview.gather('UB').result)
                UW.append(dview.gather('UW').result)
            print 'Full integration took %d seconds' % (time() - start)

        if run:
            UMF = []
            UMS = []
            UB = []
            UW = []
            run_simulation()
```

The following two cells are present to allow saving and loading of the data.

```
In [ ]: if run:
            np.savez('numeric-bkw-%d-%d-%d-%s%d.npz' %
                     (dSize, qcSize, qrdSize, filevarname, increments),
                     d=d, qc=qc, qrd=qrd,
                     UMF=UMF, UMS=UMS, UB=UB, UW=UW)
```

```
In [ ]: if not run:
            with np.load('numeric-bkw-%d-%d-%d-%s%d.npz' %
                         (dSize, qcSize, qrdSize, filevarname, increments)) as a:
                d = a['d']
                qc = a['qc']
```

```
        qrd = a['qrd']
        UMF = a['UMF']
        UMS = a['UMS']
        UB = a['UB']
        UW = a['UW']
```

From the returned energy components, we can calculate the total energy:

```
In [ ]: USphere = [2*UMF[i] + UMS[i] + UB[i] for i in range(0, increments)]
        UTot = [USphere[i] + UW[i] for i in range(0, increments)]
```

Physical behaviour

We want to see how our model behaves, and the most intuitive way to do so is to plot the compression behaviour of the knoppy web.

```
In [ ]: plot_range = [-0.8, 0.8, -0.55, 0.55]

        def interact_simple_kw(index, level):
            j = level - 1
            dVal = dVals[j]

            qcMin = 0
            qrdMin = 0
            iMin = 0
            kMin = 0
            UTmin = np.amax(UTot[index])
            UMFmin = np.amax(UMF[index])
            UMSmin = np.amax(UMS[index])
            UBmin = np.amax(UB[index])
            UWmin = np.amax(UW[index])
            for i in np.arange(qcSize):
                for k in np.arange(qrdSize):
                    if UTot[index][i, j, k] < UTmin:
                        UTmin = UTot[index][i, j, k]
                        UMFmin = UMF[index][i, j, k]
                        UMSmin = UMS[index][i, j, k]
                        UBmin = UB[index][i, j, k]
                        UWmin = UW[index][i, j, k]
                        qcMin = qcVals[i]
                        qrdMin = qrdVals[k]
                        iMin = i
                        kMin = k
            cMin = cVal(qcMin, dVal)
            rdMin = rdVal(qrdMin, cMin)

            rdd = float(ss.rdd.subs({ss.r0: r0, ss.c: cMin}))
            t = float(kw.t.subs({kw.t0: t0, kw.d: dVal, ss.c: cMin}))
            web_width = 2*(r0 + t0)
            web_height = 2*(rdd + t)
```

```
            p.plot_basic_kw_unitcell(rdMin, rdd, h, web_width, web_height, plot_range,
                                     (UMFmin, UMSmin, UBmin, UWmin))

            print "d   = %f" % dVal
            print "r'  = %f" % rdMin
            print "r'' = %f" % rdd
            print "t   = %f" % t
```

```
In [ ]: %matplotlib inline
        interact(interact_simple_kw, index=(0, increments-1), level=(1,dSize))
```

Analysis

We now plot the figures used in the thesis.

First, a surface and contour plot of U_{Tot} (figs. 4.19 and 4.21, as well as figs. 4.15a and 4.15b for the initial parameters):

```
In [ ]: # TWEAK-START Change to generate figures (b) and (c)
        i = 0
        # TWEAK-END

        vec_cVal = np.vectorize(cVal)
        dMin = d.min(2)
        cMin = vec_cVal(qc.min(2), dMin)
        clim = (np.amin(cMin), np.amax(cMin))

        p.plotSurface(dMin, cMin, UTot[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Tot}$ (gf.cm)',
                      xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, UTot[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      zlabel=r'$U_{Tot}$ (gf.cm)',
                      logNorm=False,
                      showMin=True, reshape=True)
```

Then U_{Sphere} (fig. 4.16):

```
In [ ]: p.plotSurface(dMin, cMin, USphere[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Sphere}$ (gf.cm)',
                      logNorm=False,
                      xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, USphere[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      zlabel=r'$U_{Sphere}$ (gf.cm)',
                      logNorm=True, showMin=True, reshape=True)
```

And U_{Web} (fig. 4.17):

```
In [ ]: p.plotSurface(dMin, cMin, UW[i].min(2),
                     '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Web}$ (gf.cm)',
                     logNorm=True,
                     xticks=[0, 0.1, 0.2, 0.3, 0.4])

        # Skip final point (causing artifact in minimum)
        p.plotContour(dMin[:,:-1], cMin[:,:-1], UW[i].min(2)[:,:-1], dlim, clim,
                     '', r'$d$ (cm)', r'$c$ (cm)',
                     zlabel=r'$U_{Web}$ (gf.cm)',
                     logNorm=True, showMin=True, reshape=True)
```

We can also show the evolution of the various energy terms as the sphere is compressed along the minimum total energy path. At each value of d we find the value of q_c for which U_{Tot} is a minimum, and then record the corresponding values of U_{Sphere} and U_{Web}.

```
In [ ]: def calc_mins(index):
            dmin = np.zeros(dSize)
            UTmin = np.empty(dSize)
            UTminUS = np.empty(dSize)
            UTminUMF = np.empty(dSize)
            UTminUMS = np.empty(dSize)
            UTminUB = np.empty(dSize)
            UTminUW = np.empty(dSize)

            UTmin[:] = np.amax(UTot[index])
            UTminUS[:] = np.amax(USphere[index])
            UTminUMF[:] = np.amax(UMF[index])
            UTminUMS[:] = np.amax(UMS[index])
            UTminUB[:] = np.amax(UB[index])
            UTminUW[:] = np.amax(UW[index])

            for i in np.arange(qcSize):
                for j in np.arange(dSize):
                    for k in np.arange(qrdSize):
                        if UTot[index][i,j,k] < UTmin[j]:
                            dmin[j] = d[i,j,k]
                            UTmin[j] = UTot[index][i,j,k]
                            UTminUS[j] = USphere[index][i,j,k]
                            UTminUMF[j] = UMF[index][i,j,k]
                            UTminUMS[j] = UMS[index][i,j,k]
                            UTminUB[j] = UB[index][i,j,k]
                            UTminUW[j] = UW[index][i,j,k]

            return dmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB, UTminUW

        results = [list(i) for i in zip(*[calc_mins(j) for j in range(0, increments)])]
        dmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB, UTminUW = tuple(results)
```

We plot U_{Tot} against d (figs. 4.18a, 4.18b, 4.20a and 4.20b, as well as fig. 4.15c):

```
In [ ]: # TWEAK-START Uncomment for single
        p.plotGraph(dVals, UTmin,
        #p.plotGraph(dVals, UTmin[0],
        # TWEAK-END
                    titles and r'Minimum $U_{Tot}$' or None,
                    r'$d$ (cm)', r'$U_{Tot}$ (gf.cm)',
                    logY=False, # TWEAK Set to True for figure (b)
                    linewidth=2)
```

And the corresponding values of U_S and U_W (figs. 4.18c, 4.18d and 4.20c to 4.20e):

```
In [ ]: p.plotGraph(dVals, UTminUS,
                    titles and r'Minimum $U_{Sphere}$' or None,
                    r'$d$ (cm)', r'$U_{Sphere}$ (gf.cm)',
                    linewidth=2)

        p.plotGraph(dVals, UTminUW,
                    titles and r'Minimum $U_{Web}$' or None,
                    r'$d$ (cm)', r'$U_{Web}$ (gf.cm)',
                    logY=False, # TWEAK Set to True for figure (e)
                    linewidth=2)
```

We can also plot the energy components together (fig. 4.15d):

```
In [ ]: p.plotGraph(dVals, [UTminUS[0], UTminUW[0]],
                    titles and 'Energy components' or None,
                    r'$d$ (cm)', 'Energy (gf.cm)',
                    logY=True,
                    linewidth=2)
```

B.4 BasicKnoppyWebUnitCellThickness.ipynb

This notebook is a slightly-modified version of the basic knoppy web unit cell model code from appendix B.3, to faciliate varying t_0.

```
In [ ]: local  = True
        run    = True
        titles = False
```

```
In [ ]: from ipyparallel import Client
        if local:
            rc = Client()
        else:
            rc = Client('path/to/ipcontroller-client.json')
        dview = rc[:]
        dview.clear()

        if local:
            # Only run if the engines are on the same machine.
            # On a different machine, make sure to copy equations/
            # and utils.py into the cwd of the engines.
```

```
            model_dir = %pwd
            dview.push({'model_dir': model_dir})
            %px import os
            %px os.chdir(model_dir)
```

```
In [ ]: from __future__ import division
        with dview.sync_imports():
            from math import pi
            import numpy as np
            from scipy.integrate import quad, dblquad
            from sympy import lambdify, symbols

            from equations import simple_sphere as ss, basic_kw as kw, vanwyk as vw
            from utils import lru_cache

        %px np = numpy
        %px ss = simple_sphere
        %px kw = basic_kw
        %px vw = vanwyk

        from ipywidgets import interact
        import matplotlib.pyplot as plt
        from time import time

        import plot as p
```

Constants

We start by defining constants for the model.

Web-specific

The general van Wyk equation has a few constants that we define here.

```
In [ ]: K = 0.001
        Ef = 3.98 * 10**7 # gf/cm^2 == 3.1 N/tex
        rho_w = 0.03 # g/cm^3
        rho_f = 1.3 # g/cm^3
```

Sphere-specific

The constants below are the same as we used in our analysis of the simple sphere model.

```
In [ ]: r0 = 0.25 # cm
        mu = 0.5
        h = 0.005 # cm
        Ek = ((0.05 * 10**8)/9.8) # gf/cm^2 == 0.05 GPa
```

General constants

Finally, we have constants specific to the knoppy web unit cell. The only one is t_0, the initial thickness of the web section.

```
In [ ]: t0 = [0.05, 0.15, 0.25] # cm
```

```
In [ ]: dview.push(dict(K=K, Ef=Ef, rho_w=rho_w, rho_f=rho_f))
        dview.push(dict(r0=r0, mu=mu, h=h, Ek=Ek))
        dview.push(dict(t0=t0))
```

```
In [ ]: %%px --local
        def create_subs_vars(i):
            ss_subs_vars = dict(zip([ss.r0, ss.mu, ss.h, ss.Ek], [r0, mu, h, Ek]))
            lee_subs_vars = dict(zip([vw.K, vw.Ef, vw.rho_w, vw.rho_f],
                                     [K, Ef, rho_w, rho_f]))
            kw_subs_vars = dict(zip([kw.t0], [t0[i]]))

            subs_vars = {}
            subs_vars.update(ss_subs_vars)
            subs_vars.update(lee_subs_vars)
            subs_vars.update(kw_subs_vars)
            return subs_vars
```

Helper methods

Next we define some helper methods.

The first helper method calculates c as a function of d and q_c.

```
In [ ]: %%px --local
        cmax = (r0-h)*7/8 # cm

        def cVal(qc, d, t0):
            tmax = t0*7/8
            if d <= cmax and d <= tmax:
                return qc*d
            elif d <= cmax and d > tmax:
                return qc*d + (1-qc)*(d-tmax)
            elif d > cmax and d <= tmax:
                return qc*cmax
            else:
                return qc*cmax + (1-qc)*(d-tmax)
```

```
In [ ]: %%px --local
        def dVal(qd, t0):
            return qd*(cmax + t0*7/8)
```

```
In [ ]: %%px --local
        def rdVal(qrd, c, t0):
            return qrd*(c+t0)
```

Integration

Below, we define the integration itself.

```
In [ ]: %%px --local
        vw_subs = dict(zip([vw.v, vw.v0], [kw.vc, kw.v0]))
```

```
class Numeric(object):
    def __init__(self, subs_vars, dVal, cVal, rdVal):
        # d(qd, t0)
        self.dVal = np.vectorize(lru_cache()(dVal))
        # c(qc, d, t0)
        self.cVal = np.vectorize(lru_cache()(cVal))
        # rd(qrd, c, t0)
        self.rdVal = np.vectorize(lru_cache()(rdVal))
        self.t0 = subs_vars[kw.t0]

        # Simple sphere

        self.UMF_int_inner = lambdify(ss.theta,
                                      ss.UMF_int_inner.subs(subs_vars))
        self.UMS_int = lambdify((ss.r, ss.theta, ss.c, ss.rd),
                                ss.UMS_int.subs(subs_vars))

        self.r_llim = ss.r_llim.subs(subs_vars)
        self.r_ulim = ss.r_ulim.subs(subs_vars)
        self.theta_c = lru_cache()(lambdify((ss.c),
                                            ss.theta_c.subs(subs_vars)))
        self.theta_ulim = ss.theta_ulim.subs(subs_vars)

        self.UMF_fac_inner = lambdify((ss.c, ss.rd),
                                      ss.UMF_fac_inner.subs(subs_vars))
        self.UMF_second_part = lambdify((ss.c, ss.rd),
                                        ss.UMF_second_part.subs(subs_vars))
        self.UMF_fac_outer = lambdify(symbols('y'),
                                      ss.UMF_fac_outer.subs(subs_vars))(0)
        self.UMS_fac = lambdify(symbols('y'), ss.UMS_fac.subs(subs_vars))(0)

        self.UB_val = lru_cache()(lambdify(ss.c, ss.UB.subs(subs_vars)))

        # van Wyk

        self.UW_val = lambdify((kw.d, ss.c, ss.rd),
                               vw.U.subs(vw_subs).subs(subs_vars))

        # Vector forms
        self.vec_UMFint = np.vectorize(self.UMFint, otypes='d')
        self.vec_UMSint = np.vectorize(self.UMSint, otypes='d')
        self.vec_UB     = np.vectorize(self.UB,     otypes='d')
        self.vec_UW     = np.vectorize(self.UW,     otypes='d')

    def UMFint(self, c, rd):
        return (self.UMF_fac_inner(c, rd) *
                quad(self.UMF_int_inner,
                     0, self.theta_c(c))[0]) + self.UMF_second_part(c, rd)
```

```
            def UMSint(self, c, rd):
                return dblquad(self.UMS_int,
                               self.theta_c(c), self.theta_ulim,
                               lambda x: self.r_llim, lambda x: self.r_ulim,
                               (c, rd))[0]

            def UB(self, c):
                return self.UB_val(c)

            def UW(self, d, c, rd):
                return self.UW_val(d, c, rd)

            def integrate(self, qd, qc, qrd):
                d = self.dVal(qd, self.t0)
                c = self.cVal(qc, d, self.t0)
                rd = self.rdVal(qrd, c, self.t0)

                UMF = self.UMF_fac_outer * self.vec_UMFint(c, rd)
                UMS = self.UMS_fac * self.vec_UMSint(c, rd)
                UB = self.vec_UB(c)

                UW = self.vec_UW(d, c, rd)

                return UMF, UMS, UB, UW
```

In [ ]:
```
%%px --local
def evaluate():
    return Numeric(create_subs_vars(i),
                   dVal, cVal, rdVal).integrate(qd, qc, qrd)
```

To perform the integration, we define the limits of integration of the knoppy web unit cell, and generate the mesh grids.

In [ ]:
```
qdlim = (0.001, 0.999)
qclim = (0.001, 0.999)
qrdlim = (0.001, 0.999)

qdSize = 200
qcSize = 200
qrdSize = 200

qdVals = np.linspace(*(qdlim + (qdSize,)))
qcVals = np.linspace(*(qclim + (qcSize,)))
qrdVals = np.linspace(*(qrdlim + (qrdSize,)))
qd, qc, qrd = np.meshgrid(qdVals, qcVals, qrdVals)
```

Now we run the integration. This is a slow step; the simple sphere part doesn't take long, but the web calculations require performing a double integral over a three-dimensional space.

```
In [ ]: def run_simulation():
            dview.scatter('qd', qd)
            dview.scatter('qc', qc)
            dview.scatter('qrd', qrd)

            start = time()
            last = start
            for i in range(0, len(t0)):
                dview.push(dict(i=i))
                %px UMF, UMS, UB, UW = evaluate()
                now = time()
                print 'Integration %d took %d seconds' % (i+1, now - last)
                last = now

                UMF.append(dview.gather('UMF').result)
                UMS.append(dview.gather('UMS').result)
                UB.append(dview.gather('UB').result)
                UW.append(dview.gather('UW').result)
            print 'Full integration took %d seconds' % (time() - start)

        if run:
            UMF = []
            UMS = []
            UB = []
            UW = []
            run_simulation()
```

The following two cells are present to allow saving and loading of the data.

```
In [ ]: if run:
            np.savez('numeric-bkw-%d-%d-%d-t0%d.npz' %
                     (qdSize, qcSize, qrdSize, len(t0)),
                     qd=qd, qc=qc, qrd=qrd,
                     UMF=UMF, UMS=UMS, UB=UB, UW=UW)
```

```
In [ ]: if not run:
            with np.load('numeric-bkw-%d-%d-%d-t0%d.npz' %
                         (qdSize, qcSize, qrdSize, len(t0))) as a:
                qd = a['qd']
                qc = a['qc']
                qrd = a['qrd']
                UMF = a['UMF']
                UMS = a['UMS']
                UB = a['UB']
                UW = a['UW']
```

From the returned energy components, we can calculate the total energy:

```
In [ ]: USphere = [2*UMF[i] + UMS[i] + UB[i] for i in range(0, len(t0))]
        UTot = [USphere[i] + UW[i] for i in range(0, len(t0))]
```

Physical behaviour

We want to see how our model behaves, and the most intuitive way to do so is to plot the compression behaviour of the knoppy web.

In [ ]:
```
plot_range = [-0.8, 0.8, -0.55, 0.55]

        def interact_simple_kw(index, level):
            j = level - 1
            qdVal = qdVals[j]

            qcMin = 0
            qrdMin = 0
            iMin = 0
            kMin = 0
            UTmin = np.amax(UTot[index])
            UMFmin = np.amax(UMF[index])
            UMSmin = np.amax(UMS[index])
            UBmin = np.amax(UB[index])
            UWmin = np.amax(UW[index])
            for i in np.arange(qcSize):
                for k in np.arange(qrdSize):
                    if UTot[index][i, j, k] < UTmin:
                        UTmin = UTot[index][i, j, k]
                        UMFmin = UMF[index][i, j, k]
                        UMSmin = UMS[index][i, j, k]
                        UBmin = UB[index][i, j, k]
                        UWmin = UW[index][i, j, k]
                        qcMin = qcVals[i]
                        qrdMin = qrdVals[k]
                        iMin = i
                        kMin = k
            dMin = dVal(qdVal, t0[index])
            cMin = cVal(qcMin, dVal)
            rdMin = rdVal(qrdMin, cMin)

            rdd = float(ss.rdd.subs({ss.r0: r0, ss.c: cMin}))
            t = float(kw.t.subs({kw.t0: t0[index], kw.d: dVal, ss.c: cMin}))
            web_width = 2*(r0 + t0[index])
            web_height = 2*(rdd + t)

            p.plot_basic_kw_unitcell(rdMin, rdd, h, web_width, web_height, plot_range,
                                     (UMFmin, UMSmin, UBmin, UWmin))

            print "d   = %f" % dVal
            print "r'  = %f" % rdMin
            print "r'' = %f" % rdd
            print "t   = %f" % t
```

In [ ]:
```
%matplotlib inline
```

```
interact(interact_simple_kw, index=(0, len(t0)-1), level=(1,dSize))
```

Analysis

We now plot the figures used in the thesis.

First, the surface and contour plots (fig. 4.23):

```
In [ ]: # TWEAK-START Change to generate figures (b) and (c)
        i = 0
        # TWEAK-END
        ticks = [
            [0, 0.05, 0.1, 0.15, 0.2, 0.25],
            [0, 0.1, 0.2, 0.3],
            [0, 0.1, 0.2, 0.3, 0.4],
        ]

        vec_dVal = np.vectorize(dVal)
        vec_cVal = np.vectorize(cVal)
        dMin = vec_dVal(qd.min(2), t0[i])
        cMin = vec_cVal(qc.min(2), dMin, t0[i])
        dlim = (np.amin(dMin), np.amax(dMin))
        clim = (np.amin(cMin), np.amax(cMin))

        p.plotSurface(dMin, cMin, UTot[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Tot}$ (gf.cm)',
                      xticks=ticks[i])

        p.plotContour(dMin, cMin, UTot[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      zlabel=r'$U_{Tot}$ (gf.cm)',
                      logNorm=True, showMin=True,
                      reshape=True, reshapeInterp='nn')
```

We can also show the evolution of the various energy terms as the sphere is compressed along the minimum total energy path. At each value of d we find the value of q_c for which U_{Tot} is a minimum, and then record the corresponding values of U_{Sphere} and U_{Web}.

```
In [ ]: def calc_mins(index):
            qdmin = np.zeros(qdSize)
            UTmin = np.empty(qdSize)
            UTminUS = np.empty(qdSize)
            UTminUMF = np.empty(qdSize)
            UTminUMS = np.empty(qdSize)
            UTminUB = np.empty(qdSize)
            UTminUW = np.empty(qdSize)

            UTmin[:] = np.amax(UTot[index])
            UTminUS[:] = np.amax(USphere[index])
            UTminUMF[:] = np.amax(UMF[index])
            UTminUMS[:] = np.amax(UMS[index])
```

```
            UTminUB[:] = np.amax(UB[index])
            UTminUW[:] = np.amax(UW[index])

            for i in np.arange(qcSize):
                for j in np.arange(qdSize):
                    for k in np.arange(qrdSize):
                        if UTot[index][i,j,k] < UTmin[j]:
                            qdmin[j] = qd[i,j,k]
                            UTmin[j] = UTot[index][i,j,k]
                            UTminUS[j] = USphere[index][i,j,k]
                            UTminUMF[j] = UMF[index][i,j,k]
                            UTminUMS[j] = UMS[index][i,j,k]
                            UTminUB[j] = UB[index][i,j,k]
                            UTminUW[j] = UW[index][i,j,k]

            return qdmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB, UTminUW

        results = [list(i) for i in zip(*[calc_mins(j) for j in range(0, len(t0))])]
        qdmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB, UTminUW = tuple(results)
```

We plot U_{Tot} against d (figs. 4.22a and 4.22b):

```
In [ ]: def plot_energies(Uvals, ylabel):
            fig = plt.figure()
            ax = fig.add_subplot(111)
            for i in range(0, len(t0)):
                ax.plot(vec_dVal(qdmin[i], t0[i]), Uvals[i], linewidth=2)
            ax.set_xlabel(r'$d$ (cm)')
            ax.set_ylabel(ylabel)
            # TWEAK-START Uncomment for figure (b)
        #     ax.set_yscale('log')
            # TWEAK-END
            plt.show()

        plot_energies(UTmin, r'$U_{Tot}$ (gf.cm)')
```

And the corresponding values of U_{Sphere} and U_{Web} (figs. 4.22c and 4.22d):

```
In [ ]: plot_energies(UTminUS, r'$U_{Sphere}$ (gf.cm)')
        plot_energies(UTminUW, r'$U_{Web}$ (gf.cm)')
```

Other views

The following views were not used in the thesis, but were useful for checking certain areas of the model.

```
In [ ]: p.plotSurface(dMin, cMin, USphere[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Sphere}$ (gf.cm)',
                      logNorm=True,
                      xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, USphere[i].min(2), dlim, clim,
```

```
                    '', r'$d$', r'$c$',
                    logNorm=True, showMin=True, reshape=True)
```

```
In [ ]: p.plotSurface(dMin, cMin, UW[i].min(2),
                    '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Web}$ (gf.cm)',
                    logNorm=True,
                    xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, UW[i].min(2), dlim, clim,
                    '', r'$d$', r'$c$',
                    logNorm=True, showMin=True, reshape=True)
```

B.5 ModifiedKnoppyWebUnitCell.ipynb

In this notebook, we implement the modified knoppy web unit cell from section 4.3.2. As described there, we construct the model equations to have the following variables:

- d, the distance by which the unit cell has been compressed.
- q_c, the ratio between sphere compression and web compression.
- r', the measure of the sphere's spread.

We integrate over these variables to obtain the energy map of the problem domain. Then we minimise over r' and the total energy U_{Tot} to finally obtain the relationship between unit cell compression and component compression.

```
In [ ]: local  = True
        run    = True
        titles = False
```

```
In [ ]: from ipyparallel import Client
        if local:
            rc = Client()
        else:
            rc = Client('path/to/ipcontroller-client.json')
        dview = rc[:]
        dview.clear()

        if local:
            # Only run if the engines are on the same machine.
            # On a different machine, make sure to copy equations/
            # and utils.py into the cwd of the engines.
            model_dir = %pwd
            dview.push({'model_dir': model_dir})
            %px import os
            %px os.chdir(model_dir)
```

```
In [ ]: from __future__ import division
        with dview.sync_imports():
            from math import pi
            import numpy as np
            from scipy.integrate import quad, dblquad
```

```
    from sympy import lambdify, symbols

    from equations import simple_sphere as ss, modified_kw as kw, vanwyk as vw
    from utils import lru_cache

%px np = numpy
%px ss = simple_sphere
%px kw = modified_kw
%px vw = vanwyk

from ipywidgets import interact
from time import time

import plot as p
```

Constants

We start by defining constants for the model.

Web-specific

The general van Wyk equation has a few constants that we define here.

In [ ]:
```
Ef = 3.98 * 10**7 # gf/cm^2 == 3.1 N/tex
rho_f = 1.3 # g/cm^3

# TWEAK-START Uncomment as appropriate
#K = 0.001
K = [0.001, 0.01, 0.1]
rho_w = 0.03 # g/cm^3
#rho_w = [0.01, 0.05, 0.1] # g/cm^3
# TWEAK-END
```

Sphere-specific

The constants below are the same as we used in our analysis of the simple sphere model.

In [ ]:
```
r0 = 0.25 # cm
mu = 0.5
h = 0.005 # cm
Ek = (0.05 * 10**8)/9.8 # gf/cm^2 == 0.05 GPa
```

General constants

Finally, we have constants specific to the knoppy web unit cell. The only one is t_0, the initial thickness of the web section.

In [ ]:
```
t0 = 0.25 # cm
```

In [ ]:
```
dview.push(dict(K=K, Ef=Ef, rho_w=rho_w, rho_f=rho_f))
dview.push(dict(r0=r0, mu=mu, h=h, Ek=Ek))
dview.push(dict(t0=t0))
```

```
In [ ]: %%px --local
        def create_subs_vars(i):
            ss_subs_vars = dict(zip(
                    [ss.r0, ss.mu, ss.h, ss.Ek],
                    [   r0,    mu,    h,    Ek]))
            # TWEAK-START Uncomment as appropriate
            lee_subs_vars = dict(zip(
                    [vw.K, vw.Ef, vw.rho_w, vw.rho_f],
                    [K[i],    Ef,    rho_w,    rho_f]))
        #    lee_subs_vars = dict(zip(
        #            [vw.K, vw.Ef, vw.rho_w, vw.rho_f],
        #            [   K,    Ef, rho_w[i],    rho_f]))
            # TWEAK-END
            kw_subs_vars = dict(zip([kw.t0], [t0]))

            subs_vars = {}
            subs_vars.update(ss_subs_vars)
            subs_vars.update(lee_subs_vars)
            subs_vars.update(kw_subs_vars)
            return subs_vars

        # TWEAK-START Uncomment as appropriate
        increments = len(K)
        #increments = len(rho_w)
        filevarname = 'K'
        #filevarname = 'rhow'
        # TWEAK-END
```

Helper methods

Next we define some helper methods.

The first helper method calculates c as a function of d and q_c.

```
In [ ]: %%px --local
        cmax = (r0-h)*7/8 # cm
        tmax = t0*7/8

        def cVal(qc, d):
            if d <= cmax and d <= tmax:
                return qc*d
            elif d <= cmax and d > tmax:
                return qc*d + (1-qc)*(d-tmax)
            elif d > cmax and d <= tmax:
                return qc*cmax
            else:
                return qc*cmax + (1-qc)*(d-tmax)
```

```
In [ ]: %%px --local
        def rdVal(qrd, c):
            return qrd*(c+t0)
```

Integration

Below, we define the integration itself.

```
In [ ]: %%px --local
        vw_outer_subs = dict(zip([vw.v, vw.v0], [kw.vc_outer, kw.v0_outer]))
        vw_inner_subs = dict(zip([vw.v, vw.v0], [kw.vc_inner, kw.v0_inner]))

        class Numeric(object):
            def __init__(self, subs_vars, cVal, rdVal):
                # c(qc, d)
                self.cVal = lru_cache()(cVal)
                # rd(qrd, c, t0)
                self.rdVal = lru_cache()(rdVal)

                # Simple sphere

                self.UMF_int_inner = lambdify(ss.theta,
                                              ss.UMF_int_inner.subs(subs_vars))
                self.UMS_int = lambdify((ss.r, ss.theta, ss.c, ss.rd),
                                        ss.UMS_int.subs(subs_vars))

                self.r_llim = ss.r_llim.subs(subs_vars)
                self.r_ulim = ss.r_ulim.subs(subs_vars)
                self.theta_c = lru_cache()(lambdify((ss.c),
                                                    ss.theta_c.subs(subs_vars)))
                self.theta_ulim = ss.theta_ulim.subs(subs_vars)

                self.UMF_fac_inner = lambdify((ss.c, ss.rd),
                                              ss.UMF_fac_inner.subs(subs_vars))
                self.UMF_second_part = lambdify((ss.c, ss.rd),
                                                ss.UMF_second_part.subs(subs_vars))
                self.UMF_fac_outer = lambdify(symbols('y'),
                                              ss.UMF_fac_outer.subs(subs_vars))(0)
                self.UMS_fac = lambdify(symbols('y'), ss.UMS_fac.subs(subs_vars))(0)

                self.UB_val = lru_cache()(lambdify(ss.c, ss.UB.subs(subs_vars)))

                # van Wyk

                self.U_outer_val = lambdify((kw.d, ss.c),
                                            vw.U.subs(vw_outer_subs).subs(subs_vars))
                self.U_inner_val = lambdify((kw.d, ss.c, ss.rd),
                                            vw.U.subs(vw_inner_subs).subs(subs_vars))

                # Vector forms
                self.vec_UMFint  = np.vectorize(self.UMFint,  otypes='d')
                self.vec_UMSint  = np.vectorize(self.UMSint,  otypes='d')
                self.vec_UB      = np.vectorize(self.UB,      otypes='d')
                self.vec_U_outer = np.vectorize(self.U_outer, otypes='d')
```

```
            self.vec_U_inner = np.vectorize(self.U_inner, otypes='d')

        def UMFint(self, d, qc, qrd):
            c = self.cVal(qc, d)
            rd = self.rdVal(qrd, c)
            return (self.UMF_fac_inner(c, rd) *
                    quad(self.UMF_int_inner,
                         0, self.theta_c(c))[0]) + self.UMF_second_part(c, rd)

        def UMSint(self, d, qc, qrd):
            c = self.cVal(qc, d)
            rd = self.rdVal(qrd, c)
            return dblquad(self.UMS_int,
                           self.theta_c(c), self.theta_ulim,
                           lambda x: self.r_llim, lambda x: self.r_ulim,
                           (c, rd))[0]

        def UB(self, d, qc):
            c = self.cVal(qc, d)
            return self.UB_val(c)

        def U_outer(self, d, qc):
            c = self.cVal(qc, d)
            return self.U_outer_val(d, c)

        def U_inner(self, d, qc, qrd):
            c = self.cVal(qc, d)
            rd = self.rdVal(qrd, c)
            return self.U_inner_val(d, c, rd)

        def integrate(self, d, qc, qrd):
            UMF = self.UMF_fac_outer * self.vec_UMFint(d, qc, qrd)
            UMS = self.UMS_fac * self.vec_UMSint(d, qc, qrd)
            UB = self.vec_UB(d, qc)

            U_outer = self.vec_U_outer(d, qc)
            U_inner = self.vec_U_inner(d, qc, qrd)

            return UMF, UMS, UB, U_outer, U_inner

    def evaluate(i):
        return Numeric(create_subs_vars(i), cVal, rdVal).integrate(d, qc, qrd)
```

To perform the integration, we define the limits of integration of the knoppy web unit cell, and generate the mesh grids.

```
In [ ]: dlim = (h, cmax+tmax)
        qclim = (0.001, 0.999)
        qrdlim = (0.001, 0.999)
```

```
dSize = 200
qcSize = 200
qrdSize = 200

dVals = np.linspace(*(dlim + (dSize,)))
qcVals = np.linspace(*(qclim + (qcSize,)))
qrdVals = np.linspace(*(qrdlim + (qrdSize,)))
d, qc, qrd = np.meshgrid(dVals, qcVals, qrdVals)
```

Now we run the integration. This is a slow step; the simple sphere part doesn't take long, but the web calculations require performing a double integral over a three-dimensional space.

```
In [ ]: def run_simulation():
            dview.scatter('d', d)
            dview.scatter('qc', qc)
            dview.scatter('qrd', qrd)

            start = time()
            last = start
            for i in range(0, increments):
                dview.push(dict(i=i))
                %px UMF, UMS, UB, U_outer, U_inner = evaluate(i)
                now = time()
                print 'Integration %d took %d seconds' % (i+1, now - last)
                last = now

                UMF.append(dview.gather('UMF').result)
                UMS.append(dview.gather('UMS').result)
                UB.append(dview.gather('UB').result)
                U_outer.append(dview.gather('U_outer').result)
                U_inner.append(dview.gather('U_inner').result)
            print 'Full integration took %d seconds' % (time() - start)

        if run:
            UMF = []
            UMS = []
            UB = []
            U_outer = []
            U_inner = []
            run_simulation()
```

The following two cells are present to allow saving and loading of the data.

```
In [ ]: if run:
            np.savez('numeric-mkw-%d-%d-%d-%s%d.npz' %
                     (dSize, qcSize, qrdSize, filevarname, increments),
                     d=d, qc=qc, qrd=qrd,
                     UMF=UMF, UMS=UMS, UB=UB,
                     U_outer=U_outer, U_inner=U_inner)
```

```
In [ ]: if not run:
            with np.load('numeric-mkw-%d-%d-%d-%s%d.npz' %
                         (dSize, qcSize, qrdSize, filevarname, increments)) as a:
                d = a['d']
                qc = a['qc']
                qrd = a['qrd']
                UMF = a['UMF']
                UMS = a['UMS']
                UB = a['UB']
                U_outer = a['U_outer']
                U_inner = a['U_inner']
```

From the returned energy components, we can calculate the total energy:

```
In [ ]: USphere = [2*UMF[i] + UMS[i] + UB[i] for i in range(0, increments)]
        UWeb = [2*U_outer[i] + U_inner[i] for i in range(0, increments)]
        UTot = [USphere[i] + UWeb[i] for i in range(0, increments)]
```

Physical behaviour

We want to see how our model behaves, and the most intuitive way to do so is to plot the compression behaviour of the knoppy web.

```
In [ ]: plot_range = [-0.8, 0.8, -0.55, 0.55]

        def interact_simple_kw(index, level):
            j = level - 1
            dVal = dVals[j]

            qcMin = 0
            qrdMin = 0
            iMin = 0
            kMin = 0
            UTmin = np.amax(UTot[index])
            UMFmin = np.amax(UMF[index])
            UMSmin = np.amax(UMS[index])
            UBmin = np.amax(UB[index])
            UouterMin = np.amax(U_outer[index])
            UinnerMin = np.amax(U_inner[index])
            for i in np.arange(qcSize):
                for k in np.arange(qrdSize):
                    if UTot[index][i, j, k] < UTmin:
                        UTmin = UTot[index][i, j, k]
                        UMFmin = UMF[index][i, j, k]
                        UMSmin = UMS[index][i, j, k]
                        UBmin = UB[index][i, j, k]
                        UouterMin = U_outer[index][i, j, k]
                        UinnerMin = U_inner[index][i, j, k]
                        qcMin = qcVals[i]
                        qrdMin = qrdVals[k]
```

```
                    iMin = i
                    kMin = k
            cMin = cVal(qcMin, dVal)
            rdMin = rdVal(qrdMin, cMin)

            rdd = float(ss.rdd.subs({ss.r0: r0, ss.c: cMin}))
            t = float(kw.t.subs({kw.t0: t0, kw.d: dVal, ss.c: cMin}))
            web_width = 2*(r0 + t0)
            web_height = 2*(rdd + t)

            p.plot_basic_kw_unitcell(rdMin, rdd, h, web_width, web_height, plot_range,
                                     (UMFmin, UMSmin, UBmin, UouterMin, UinnerMin))

            print "d   = %f" % dVal
            print "r'  = %f" % rdMin
            print "r'' = %f" % rdd
            print "t   = %f" % t
```

```
In [ ]: %matplotlib inline
        interact(interact_simple_kw, index=(0, increments-1), level=(1,dSize))
```

Analysis

We now plot the figures used in the thesis.

First, a surface and contour plot of U_{Tot} (figs. 4.29 and 4.31, as well as figures figs. 4.25a and 4.25b for the initial parameters):

```
In [ ]: # TWEAK-START Change to generate figures (b) and (c)
        i = 0
        # TWEAK-END

        vec_cVal = np.vectorize(cVal)
        dMin = d.min(2)
        cMin = vec_cVal(qc.min(2), dMin)
        clim = (np.amin(cMin), np.amax(cMin))

        p.plotSurface(dMin, cMin, UTot[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Tot}$ (gf.cm)',
                      xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, UTot[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      zlabel=r'$U_{Tot}$ (gf.cm)',
                      logNorm=False,
                      showMin=True, reshape=True)
```

Then U_{Sphere} (fig. 4.26):

```
In [ ]: p.plotSurface(dMin, cMin, USphere[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Sphere}$ (gf.cm)',
                      logNorm=True,
```

```
                    xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, USphere[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      zlabel=r'$U_{Sphere}$ (gf.cm)',
                      logNorm=True, showMin=True, reshape=True)
```

And U_{Web} (fig. 4.27):

```
In [ ]: p.plotSurface(dMin, cMin, UWeb[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Web}$ (gf.cm)',
                      logNorm=True,
                      xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, UWeb[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      zlabel=r'$U_{Web}$ (gf.cm)',
                      logNorm=True, showMin=True, reshape=True)
```

We can also show the evolution of the various energy terms as the sphere is compressed along the minimum total energy path. At each value of d we find the value of q_c for which U_{Tot} is a minimum, and then record the corresponding values of U_{Sphere} and U_{Web}.

```
In [ ]: def calc_mins(index):
            dmin = np.zeros(dSize)
            UTmin = np.empty(dSize)
            UTminUS = np.empty(dSize)
            UTminUMF = np.empty(dSize)
            UTminUMS = np.empty(dSize)
            UTminUB = np.empty(dSize)
            UTminUW = np.empty(dSize)
            UTminUWo = np.empty(dSize)
            UTminUWi = np.empty(dSize)

            UTmin[:] = np.amax(UTot[index])
            UTminUS[:] = np.amax(USphere[index])
            UTminUMF[:] = np.amax(UMF[index])
            UTminUMS[:] = np.amax(UMS[index])
            UTminUB[:] = np.amax(UB[index])
            UTminUW[:] = np.amax(UWeb[index])
            UTminUWo[:] = np.amax(U_outer[index])
            UTminUWi[:] = np.amax(U_inner[index])

            for i in np.arange(qcSize):
                for j in np.arange(dSize):
                    for k in np.arange(qrdSize):
                        if UTot[index][i,j,k] < UTmin[j]:
                            dmin[j] = d[i,j,k]
                            UTmin[j] = UTot[index][i,j,k]
                            UTminUS[j] = USphere[index][i,j,k]
```

```
                            UTminUMF[j] = UMF[index][i,j,k]
                            UTminUMS[j] = UMS[index][i,j,k]
                            UTminUB[j] = UB[index][i,j,k]
                            UTminUW[j] = UWeb[index][i,j,k]
                            UTminUWo[j] = U_outer[index][i,j,k]
                            UTminUWi[j] = U_inner[index][i,j,k]

            return (dmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB,
                    UTminUW, UTminUWo, UTminUWi)

        results = [list(i) for i in zip(*[
                    calc_mins(j) for j in range(0, increments)])]
        dmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB, \
              UTminUW, UTminUWo, UTminUWi = tuple(results)
```

We plot U_{Tot} against d (figs. 4.28a, 4.28b, 4.30a and 4.30b, as well as fig. 4.25c):

```
In [ ]: # TWEAK-START Uncomment for single
        p.plotGraph(dVals, UTmin,
        #p.plotGraph(dVals, UTmin[0],
        # TWEAK-END
                    titles and r'Minimum $U_{Tot}$' or None,
                    r'$d$ (cm)', r'$U_{Tot}$ (gf.cm)',
                    logY=False, # TWEAK Set to True for figure (b)
                    linewidth=2)
```

And the corresponding values of U_{Sphere} and U_{Web} (figs. 4.28c, 4.28d and 4.30c to 4.30e):

```
In [ ]: p.plotGraph(dVals, UTminUS,
                    titles and r'Minimum $U_{Sphere}$' or None,
                    r'$d$ (cm)', r'$U_{Sphere}$ (gf.cm)',
                    linewidth=2)

        p.plotGraph(dVals, UTminUW,
                    titles and r'Minimum $U_{Web}$' or None,
                    r'$d$ (cm)', r'$U_{Web}$ (gf.cm)',
                    logY=False, # TWEAK Set to True for figure (e)
                    linewidth=2)
```

We can also plot the energy components together (fig. 4.25d):

```
In [ ]: p.plotGraph(dVals, [UTminUS[0], UTminUW[0]],
                    titles and 'Energy components' or None,
                    r'$d$ (cm)', 'Energy (gf.cm)',
                    logY=True,
                    linewidth=2)
```

B.6 ModifiedKnoppyWebUnitCellThickness.ipynb

This notebook is a slightly-modified version of the basic knoppy web unit cell model code from appendix B.5, to faciliate varying t_0.

```
In [ ]: local  = True
        run    = True
        titles = False

In [ ]: from ipyparallel import Client
        if local:
            rc = Client()
        else:
            rc = Client('path/to/ipcontroller-client.json')
        dview = rc[:]
        dview.clear()

        if local:
            # Only run if the engines are on the same machine.
            # On a different machine, make sure to copy equations/
            # and utils.py into the cwd of the engines.
            model_dir = %pwd
            dview.push({'model_dir': model_dir})
            %px import os
            %px os.chdir(model_dir)

In [ ]: from __future__ import division
        with dview.sync_imports():
            from math import pi
            import numpy as np
            from scipy.integrate import quad, dblquad
            from sympy import lambdify, symbols

            from equations import simple_sphere as ss, modified_kw as kw, vanwyk as vw
            from utils import lru_cache

        %px np = numpy
        %px ss = simple_sphere
        %px kw = modified_kw
        %px vw = vanwyk

        from ipywidgets import interact
        import matplotlib.pyplot as plt
        from time import time

        import plot as p
```

Constants

We start by defining constants for the model.

Web-specific

The general van Wyk equation has a few constants that we define here.

```
In [ ]: K = 0.001
        Ef = 3.98 * 10**7 # gf/cm^2 == 3.1 N/tex
        rho_w = 0.03 # g/cm^3
        rho_f = 1.3 # g/cm^3
```

Sphere-specific

The constants below are the same as we used in our analysis of the simple sphere model.

```
In [ ]: r0 = 0.25 # cm
        mu = 0.5
        h = 0.005 # cm
        Ek = (0.05 * 10**8)/9.8 # gf/cm^2 == 0.05 GPa
```

General constants

Finally, we have constants specific to the knoppy web unit cell. The only one is t_0, the initial thickness of the web section.

```
In [ ]: t0 = [0.05, 0.15, 0.25] # cm
```

```
In [ ]: dview.push(dict(K=K, Ef=Ef, rho_w=rho_w, rho_f=rho_f))
        dview.push(dict(r0=r0, mu=mu, h=h, Ek=Ek))
        dview.push(dict(t0=t0))
```

```
In [ ]: %%px --local
        def create_subs_vars(i):
            ss_subs_vars = dict(zip([ss.r0, ss.mu, ss.h, ss.Ek], [r0, mu, h, Ek]))
            lee_subs_vars = dict(zip([vw.K, vw.Ef, vw.rho_w, vw.rho_f],
                                     [K, Ef, rho_w, rho_f]))
            kw_subs_vars = dict(zip([kw.t0], [t0[i]]))

            subs_vars = {}
            subs_vars.update(ss_subs_vars)
            subs_vars.update(lee_subs_vars)
            subs_vars.update(kw_subs_vars)
            return subs_vars
```

Helper methods

Next we define some helper methods.

The first helper method calculates c as a function of d and q_c.

```
In [ ]: %%px --local
        cmax = (r0-h)*7/8 # cm

        def cVal(qc, d, t0):
            tmax = t0*7/8
            if d <= cmax and d <= tmax:
                return qc*d
            elif d <= cmax and d > tmax:
                return qc*d + (1-qc)*(d-tmax)
```

```
            elif d > cmax and d <= tmax:
                return qc*cmax
            else:
                return qc*cmax + (1-qc)*(d-tmax)
```

```
In [ ]: %%px --local
        def dVal(qd, t0):
            return qd*(cmax + t0*7/8)
```

```
In [ ]: %%px --local
        def rdVal(qrd, c, t0):
            return qrd*(c+t0)
```

Integration

Below, we define the integration itself.

```
In [ ]: %%px --local
        vw_outer_subs = dict(zip([vw.v, vw.v0], [kw.vc_outer, kw.v0_outer]))
        vw_inner_subs = dict(zip([vw.v, vw.v0], [kw.vc_inner, kw.v0_inner]))

        class Numeric(object):
            def __init__(self, subs_vars, dVal, cVal, rdVal):
                # d(qd, t0)
                self.dVal = np.vectorize(lru_cache()(dVal))
                # c(qc, d, t0)
                self.cVal = np.vectorize(lru_cache()(cVal))
                # rd(qrd, c, t0)
                self.rdVal = np.vectorize(lru_cache()(rdVal))
                self.t0 = subs_vars[kw.t0]

                # Simple sphere

                self.UMF_int_inner = lambdify(ss.theta,
                                              ss.UMF_int_inner.subs(subs_vars))
                self.UMS_int = lambdify((ss.r, ss.theta, ss.c, ss.rd),
                                        ss.UMS_int.subs(subs_vars))

                self.r_llim = ss.r_llim.subs(subs_vars)
                self.r_ulim = ss.r_ulim.subs(subs_vars)
                self.theta_c = lru_cache()(lambdify((ss.c),
                                                    ss.theta_c.subs(subs_vars)))
                self.theta_ulim = ss.theta_ulim.subs(subs_vars)

                self.UMF_fac_inner = lambdify((ss.c, ss.rd),
                                              ss.UMF_fac_inner.subs(subs_vars))
                self.UMF_second_part = lambdify((ss.c, ss.rd),
                                                ss.UMF_second_part.subs(subs_vars))
                self.UMF_fac_outer = lambdify(symbols('y'),
                                              ss.UMF_fac_outer.subs(subs_vars))(0)
```

```
        self.UMS_fac = lambdify(symbols('y'), ss.UMS_fac.subs(subs_vars))(0)

        self.UB_val = lru_cache()(lambdify(ss.c, ss.UB.subs(subs_vars)))

        # van Wyk

        self.U_outer_val = lambdify((kw.d, ss.c),
                                    vw.U.subs(vw_outer_subs).subs(subs_vars))
        self.U_inner_val = lambdify((kw.d, ss.c, ss.rd),
                                    vw.U.subs(vw_inner_subs).subs(subs_vars))

        # Vector forms
        self.vec_UMFint  = np.vectorize(self.UMFint,  otypes='d')
        self.vec_UMSint  = np.vectorize(self.UMSint,  otypes='d')
        self.vec_UB      = np.vectorize(self.UB,      otypes='d')
        self.vec_U_outer = np.vectorize(self.U_outer, otypes='d')
        self.vec_U_inner = np.vectorize(self.U_inner, otypes='d')

    def UMFint(self, c, rd):
        return (self.UMF_fac_inner(c, rd) *
                quad(self.UMF_int_inner,
                     0, self.theta_c(c))[0]) + self.UMF_second_part(c, rd)

    def UMSint(self, c, rd):
        return dblquad(self.UMS_int,
                       self.theta_c(c), self.theta_ulim,
                       lambda x: self.r_llim, lambda x: self.r_ulim,
                       (c, rd))[0]

    def UB(self, c):
        return self.UB_val(c)

    def U_outer(self, d, c):
        return self.U_outer_val(d, c)

    def U_inner(self, d, c, rd):
        return self.U_inner_val(d, c, rd)

    def integrate(self, qd, qc, qrd):
        d = self.dVal(qd, self.t0)
        c = self.cVal(qc, d, self.t0)
        rd = self.rdVal(qrd, c, self.t0)

        UMF = self.UMF_fac_outer * self.vec_UMFint(c, rd)
        UMS = self.UMS_fac * self.vec_UMSint(c, rd)
        UB = self.vec_UB(c)

        U_outer = self.vec_U_outer(d, c)
        U_inner = self.vec_U_inner(d, c, rd)
```

```
            return UMF, UMS, UB, U_outer, U_inner

        def evaluate(i):
            return Numeric(create_subs_vars(i),
                           dVal, cVal, rdVal).integrate(qd, qc, qrd)
```

To perform the integration, we define the limits of integration of the knoppy web unit cell, and generate the mesh grids.

```
In [ ]: qdlim = (0.001, 0.999)
        qclim = (0.001, 0.999)
        qrdlim = (0.001, 0.999)

        qdSize = 200
        qcSize = 200
        qrdSize = 200

        qdVals = np.linspace(*(qdlim + (qdSize,)))
        qcVals = np.linspace(*(qclim + (qcSize,)))
        qrdVals = np.linspace(*(qrdlim + (qrdSize,)))
        qd, qc, qrd = np.meshgrid(qdVals, qcVals, qrdVals)
```

Now we run the integration. This is a slow step; the simple sphere part doesn't take long, but the web calculations require performing a double integral over a three-dimensional space.

```
In [ ]: def run_simulation():
            dview.scatter('qd', qd)
            dview.scatter('qc', qc)
            dview.scatter('qrd', qrd)

            start = time()
            last = start
            for i in range(0, len(t0)):
                dview.push(dict(i=i))
                %px UMF, UMS, UB, U_outer, U_inner = evaluate(i)
                now = time()
                print 'Integration %d took %d seconds' % (i+1, now - last)
                last = now

                UMF.append(dview.gather('UMF').result)
                UMS.append(dview.gather('UMS').result)
                UB.append(dview.gather('UB').result)
                U_outer.append(dview.gather('U_outer').result)
                U_inner.append(dview.gather('U_inner').result)
            print 'Full integration took %d seconds' % (time() - start)

        if run:
            UMF = []
            UMS = []
```

```
            UB = []
            U_outer = []
            U_inner = []
            run_simulation()
```

The following two cells are present to allow saving and loading of the data.

```
In [ ]: if run:
            np.savez('numeric-mkw-%d-%d-%d-t0%d.npz' %
                     (qdSize, qcSize, qrdSize, len(t0)),
                     qd=qd, qc=qc, qrd=qrd,
                     UMF=UMF, UMS=UMS, UB=UB,
                     U_outer=U_outer, U_inner=U_inner)
```

```
In [ ]: if not run:
            with np.load('numeric-mkw-%d-%d-%d-t0%d.npz' %
                         (qdSize, qcSize, qrdSize, len(t0))) as a:
                qd = a['qd']
                qc = a['qc']
                qrd = a['qrd']
                UMF = a['UMF']
                UMS = a['UMS']
                UB = a['UB']
                U_outer = a['U_outer']
                U_inner = a['U_inner']
```

From the returned energy components, we can calculate the total energy:

```
In [ ]: USphere = [2*UMF[i] + UMS[i] + UB[i] for i in range(0, len(t0))]
        UWeb = [2*U_outer[i] + U_inner[i] for i in range(0, len(t0))]
        UTot = [USphere[i] + UWeb[i] for i in range(0, len(t0))]
```

Physical behaviour

We want to see how our model behaves, and the most intuitive way to do so is to plot the compression behaviour of the knoppy web.

```
In [ ]: plot_range = [-0.8, 0.8, -0.55, 0.55]

        def interact_simple_kw(index, level):
            j = level - 1
            qdVal = qdVals[j]

            qcMin = 0
            qrdMin = 0
            iMin = 0
            kMin = 0
            UTmin = np.amax(UTot[index])
            UMFmin = np.amax(UMF[index])
            UMSmin = np.amax(UMS[index])
            UBmin = np.amax(UB[index])
```

```
            UouterMin = np.amax(U_outer[index])
            UinnerMin = np.amax(U_inner[index])
            for i in np.arange(qcSize):
                for k in np.arange(qrdSize):
                    if UTot[index][i, j, k] < UTmin:
                        UTmin = UTot[index][i, j, k]
                        UMFmin = UMF[index][i, j, k]
                        UMSmin = UMS[index][i, j, k]
                        UBmin = UB[index][i, j, k]
                        UouterMin = U_outer[index][i, j, k]
                        UinnerMin = U_inner[index][i, j, k]
                        qcMin = qcVals[i]
                        qrdMin = qrdVals[k]
                        iMin = i
                        kMin = k
            dMin = dVal(qdVal, t0[index])
            cMin = cVal(qcMin, dMin, t0[index])
            rdMin = rdVal(qrdMin, cMin, t0[index])

            rdd = float(ss.rdd.subs({ss.r0: r0, ss.c: cMin}))
            t = float(kw.t.subs({kw.t0: t0[index], kw.d: dMin, ss.c: cMin}))
            web_width = 2*(r0 + t0[index])
            web_height = 2*(rdd + t)

            p.plot_basic_kw_unitcell(rdMin, rdd, h, web_width, web_height, plot_range,
                                     (UMFmin, UMSmin, UBmin, UouterMin, UinnerMin))

            print "d   = %f" % dMin
            print "r'  = %f" % rdMin
            print "r'' = %f" % rdd
            print "t   = %f" % t
```

```
In [ ]: %matplotlib inline
        interact(interact_simple_kw, index=(0, len(t0)-1), level=(1,qdSize))
```

Analysis

We now plot the figures used in the thesis.

First, the surface and contour plots (fig. 4.33):

```
In [ ]: # TWEAK-START Change to generate figures (b) and (c)
        i = 0
        # TWEAK-END
        ticks = [
            [0, 0.05, 0.1, 0.15, 0.2, 0.25],
            [0, 0.1, 0.2, 0.3],
            [0, 0.1, 0.2, 0.3, 0.4],
        ]

        vec_dVal = np.vectorize(dVal)
```

```
        vec_cVal = np.vectorize(cVal)
        dMin = vec_dVal(qd.min(2), t0[i])
        cMin = vec_cVal(qc.min(2), dMin, t0[i])
        dlim = (np.amin(dMin), np.amax(dMin))
        clim = (np.amin(cMin), np.amax(cMin))

        p.plotSurface(dMin, cMin, UTot[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Tot}$ (gf.cm)',
                      xticks=ticks[i])

        p.plotContour(dMin, cMin, UTot[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      zlabel=r'$U_{Tot}$ (gf.cm)',
                      logNorm=True, showMin=True,
                      reshape=True, reshapeInterp='nn')
```

We can also show the evolution of the various energy terms as the sphere is compressed along the minimum total energy path. At each value of d we find the value of q_c for which U_{Tot} is a minimum, and then record the corresponding values of U_{Sphere} and U_{Web}.

```
In [ ]: def calc_mins(index):
            qdmin = np.zeros(qdSize)
            UTmin = np.empty(qdSize)
            UTminUS = np.empty(qdSize)
            UTminUMF = np.empty(qdSize)
            UTminUMS = np.empty(qdSize)
            UTminUB = np.empty(qdSize)
            UTminUW = np.empty(qdSize)
            UTminUWo = np.empty(qdSize)
            UTminUWi = np.empty(qdSize)

            UTmin[:] = np.amax(UTot[index])
            UTminUS[:] = np.amax(USphere[index])
            UTminUMF[:] = np.amax(UMF[index])
            UTminUMS[:] = np.amax(UMS[index])
            UTminUB[:] = np.amax(UB[index])
            UTminUW[:] = np.amax(UWeb[index])
            UTminUWo[:] = np.amax(U_outer[index])
            UTminUWi[:] = np.amax(U_inner[index])

            for i in np.arange(qcSize):
                for j in np.arange(qdSize):
                    for k in np.arange(qrdSize):
                        if UTot[index][i,j,k] < UTmin[j]:
                            qdmin[j] = qd[i,j,k]
                            UTmin[j] = UTot[index][i,j,k]
                            UTminUS[j] = USphere[index][i,j,k]
                            UTminUMF[j] = UMF[index][i,j,k]
                            UTminUMS[j] = UMS[index][i,j,k]
```

```
                UTminUB[j] = UB[index][i,j,k]
                UTminUW[j] = UWeb[index][i,j,k]
                UTminUWo[j] = U_outer[index][i,j,k]
                UTminUWi[j] = U_inner[index][i,j,k]

        return (qdmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB,
                UTminUW, UTminUWo, UTminUWi)

    results = [list(i) for i in zip(*[
                calc_mins(j) for j in range(0, len(t0))])]
    qdmin, UTmin, UTminUS, UTminUMF, UTminUMS, UTminUB, \
           UTminUW, UTminUWo, UTminUWi = tuple(results)
```

We plot U_{Tot} against d (figs. 4.32a and 4.32b):

```
In [ ]: def plot_energies(Uvals, ylabel):
            fig = plt.figure()
            ax = fig.add_subplot(111)
            for i in range(0, len(t0)):
                ax.plot(vec_dVal(qdmin[i], t0[i]), Uvals[i], linewidth=2)
            ax.set_xlabel(r'$d$ (cm)')
            ax.set_ylabel(ylabel)
            # TWEAK-START Uncomment for figure (b)
        #    ax.set_yscale('log')
            # TWEAK-END
            plt.show()

        plot_energies(UTmin, r'$U_{Tot}$ (gf.cm)')
```

And the corresponding values of U_{Sphere} and U_{Web} (figs. 4.32c and 4.32d):

```
In [ ]: plot_energies(UTminUS, r'$U_{Sphere}$ (gf.cm)')
        plot_energies(UTminUW, r'$U_{Web}$ (gf.cm)')
```

Other views

The following views were not used in the thesis, but were useful for checking certain areas of the model.

```
In [ ]: p.plotSurface(dMin, cMin, USphere[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Sphere}$ (gf.cm)',
                      logNorm=True,
                      xticks=[0, 0.1, 0.2, 0.3, 0.4])

        p.plotContour(dMin, cMin, USphere[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      logNorm=True, showMin=True, reshape=True)

In [ ]: p.plotSurface(dMin, cMin, UWeb[i].min(2),
                      '', r'$d$ (cm)', r'$c$ (cm)', r'$U_{Web}$ (gf.cm)',
                      logNorm=True,
                      xticks=[0, 0.1, 0.2, 0.3, 0.4])
```

```
        p.plotContour(dMin, cMin, UWeb[i].min(2), dlim, clim,
                      '', r'$d$ (cm)', r'$c$ (cm)',
                      logNorm=True, showMin=True, reshape=True)
```

B.7 SimpleSphereFit.ipynb

In this notebook, we implement the simple knoppy web model from section 4.2 of the thesis, in a way that can be fitted to experimental data.

We start with imports, as usual.

```
In [ ]: from __future__ import division
        import numpy as np
        from scipy.integrate import quad, dblquad
        from sympy import lambdify

        from equations import simple_sphere as ss
        from utils import lru_cache

        from lmfit import Model
        from time import time

        import matplotlib.pyplot as plt
        import plot as p
```

Data

We list the data available:

```
In [ ]: # Filename, r0 + h
        data = [
            ('K Lots', 0.366),
            ('K CR Bonded', 0.383),
        ]
```

Fitting the model to data

Now we define the model itself.

```
In [ ]: class Numeric(object):
            def __init__(self, subs_vars):
                self.UMF_int_inner = lambdify(ss.theta, ss.UMF_int_inner)
                self.UMS_int = lambdify((ss.r, ss.theta, ss.c),
                                        ss.UMS_int.subs(subs_vars))

                self.r_llim = ss.r_llim.subs(subs_vars)
                self.r_ulim = ss.r_ulim.subs(subs_vars)
                self.theta_c = lru_cache()(lambdify((ss.c),
                                                    ss.theta_c.subs(subs_vars)))
                self.theta_ulim = ss.theta_ulim.subs(subs_vars)
```

```
        self.UMF_fac_inner = np.vectorize(lru_cache()(
                lambdify((ss.c),
                         ss.UMF_fac_inner.subs(subs_vars))))
        self.UMF_second_part = np.vectorize(
                lambdify((ss.c, ss.mu),
                         ss.UMF_second_part.subs(subs_vars)))
        self.UMF_fac_outer = lru_cache()(lambdify((ss.mu, ss.Ek),
                                    ss.UMF_fac_outer.subs(subs_vars)))
        self.UMS_fac = lru_cache()(lambdify((ss.Ek),
                                ss.UMS_fac.subs(subs_vars)))

        self.UB = lru_cache()(lambdify((ss.c, ss.Ek),
                                     ss.UB.subs(subs_vars)))

        self.vec_UMFint = np.vectorize(self.UMFint)
        self.vec_UMSint = np.vectorize(self.UMSint)
        self.vec_UB = np.vectorize(self.UB)

    def UMFint(self, c):
        return quad(self.UMF_int_inner, 0, self.theta_c(c))[0]

    def UMSint(self, c):
        return dblquad(self.UMS_int,
                       self.theta_c(c), self.theta_ulim,
                       lambda x: self.r_llim, lambda x: self.r_ulim,
                       (c,))[0]

    def prepare(self, c):
        self.UMFint_val = self.vec_UMFint(c)
        self.UMSint_val = self.vec_UMSint(c)

    def integrate(self, c, mu, Ek):
        UMF = self.UMF_fac_outer(mu, Ek) * (self.UMF_fac_inner(c) *
                                    self.UMFint_val +
                                    self.UMF_second_part(c, mu))
        UMS = self.UMS_fac(Ek) * self.UMSint_val
        UB = self.vec_UB(c, Ek)
        return UMF, UMS, UB

def get_numeric_inner(c, h):
    subs_vars = dict(zip(
            [ss.rd, ss.r0,        ss.h],
            [ss.c,  data[i][1]-h, h]
        ))
    n = Numeric(subs_vars)
    n.prepare(c)
    return n
get_numeric = lru_cache()(get_numeric_inner)
```

```
        def run_model(c, h, mu, Ek):
            n = get_numeric(c, h)
            return n.integrate(c, mu, Ek)

        def f(c, h, mu, Ek):
            # Convert normalised c to model scale
            UMF, UMS, UB = run_model(c*data[i][1], h, mu, Ek)
            # Return normalised energy for fitting
            return (2*UMF + UMS + UB)/((2*data[i][1])**3)

        model = Model(f, independent_vars=['c'])
```

Then we import the data and define initial guesses.

```
In [ ]: def load_data(i):
            c, UT = np.loadtxt('k-compression-csv/%s.csv' % data[i][0],
                               delimiter=',', skiprows=1,
                               usecols = (5, 6), unpack=True)

            c = c[1:]
            UT = UT[1:]

            params = model.make_params()
            params['h'].value = 0.005
            params['h'].min = 0.001
            params['h'].max = data[i][1] * (1-c[-1]) / 2
            params['mu'].value = 0.4
            params['mu'].min = 0
            params['mu'].max = 0.5
            params['Ek'].value = 100 # gf/cm^2
            params['Ek'].min = 0

            return c, UT, params
```

Finally, we perform the fitting itself.

```
In [ ]: results = []
        for i in range(0, len(data)):
            c, UT, params = load_data(i)
            start = time()
            results.append(model.fit(UT, params, c=c))
            print 'Fitting took %d seconds' % (time() - start)

            with open('k-compression-csv/result-sim-%s.txt' %
                      data[i][0], 'w') as text_file:
                text_file.write(results[i].fit_report())
```

And we plot the fit alongside the data (figure 5.43).

```
In [ ]: # TWEAK-START Set to 1 for figure (b)
        i = 0
```

```
        # TWEAK-END

        print results[i].fit_report()

        c, UT, _ = load_data(i)

        plt.plot(c, UT,         'b', linewidth=2)
        plt.plot(c, results[i].init_fit, 'k--', linewidth=2)
        plt.plot(c, results[i].best_fit, 'r-', linewidth=1.5)

        ax = plt.gca()
        ax.set_xlabel('Normalised displacement (cm/cm)')
        ax.set_ylabel(r'Energy density (gf.cm/cm$^3$)')
        # TWEAK-START Uncomment to get log plots
        #ax.set_yscale('log')
        # TWEAK-END

        plt.show()
```

Other views

The following views were not used in the thesis, but were useful for checking certain areas of the model.

```
In [ ]: def plot_parts(values):
            h = values['h']
            r0 = data[i][1]-h
            vol = ((2*(r0+h))**3)

            UMF, UMS, UB = run_model(c*r0, **values)
            UMF = UMF/vol
            UMS = UMS/vol
            UB = UB/vol

            plt.plot(c, UT, 'k--', linewidth=2)
            plt.plot(c, 2*UMF, 'b', linewidth=2)
            plt.plot(c, UMS, 'r', linewidth=2)
            plt.plot(c, UB, 'g', linewidth=2)

            ax = plt.gca()
            ax.set_xlabel('Normalised displacement (cm/cm)')
            ax.set_ylabel(r'Energy density (gf.cm/cm$^3$)')
            ax.set_yscale('log')

            plt.show()

        plot_parts(results[i].values)
```

B.8 FibrousSimpleSphereFit.ipynb

In this notebook, we implement a variant of the simple knoppy web model from section 4.2, in a way that can be fitted to experimental data. The variations include:

- Restricting the knop from expanding at all (ie. set $r' = c$)
- Filling the space around the sphere with van Wyk fibrous web

These modifications allow the model to be fitted to the knop compression data, and accounts for the fact that factory-produced knops do have a fraction of non-knopped fibre. The fraction is over-represented by volume in this model (the excess volume is 47.6% of the cubic space surrounding the sphere), but this will also be the case in the knoppy web model.

We start with imports, as usual.

In [ ]:
```
from __future__ import division
import numpy as np
from scipy.integrate import quad, dblquad
from sympy import lambdify

from equations import simple_sphere as ss, basic_kw as kw, vanwyk as vw
from utils import lru_cache

from lmfit import Model
from time import time

import matplotlib.pyplot as plt
import plot as p
```

In [ ]:
```
Ef = 3.98 * 10**7 # gf/cm^2 == 3.1 N/tex
rho_f = 1.3 # g/cm^3
```

Data

We list the data available:

In [ ]:
```
# Filename, r0 + h
data = [
    ('K Lots', 0.366),
    ('K CR Bonded', 0.383),
]
```

Fitting the model to data

Now we define the model itself.

In [ ]:
```
vw_subs = dict(zip([vw.v, vw.v0], [kw.vc, kw.v0]))

class Numeric(object):
    def __init__(self, subs_vars):
        # Simple sphere
        self.UMF_int_inner = lambdify(ss.theta, ss.UMF_int_inner)
        self.UMS_int = lambdify((ss.r, ss.theta, ss.c),
```

```
                                ss.UMS_int.subs(subs_vars))

        self.r_llim = ss.r_llim.subs(subs_vars)
        self.r_ulim = ss.r_ulim.subs(subs_vars)
        self.theta_c = lru_cache()(lambdify((ss.c),
                                            ss.theta_c.subs(subs_vars)))
        self.theta_ulim = ss.theta_ulim.subs(subs_vars)

        self.UMF_fac_inner = np.vectorize(lru_cache()(
                lambdify((ss.c),
                         ss.UMF_fac_inner.subs(subs_vars))))
        self.UMF_second_part = np.vectorize(
                lambdify((ss.c, ss.mu),
                         ss.UMF_second_part.subs(subs_vars)))
        self.UMF_fac_outer = lru_cache()(lambdify((ss.mu, ss.Ek),
                                         ss.UMF_fac_outer.subs(subs_vars)))
        self.UMS_fac = lru_cache()(lambdify((ss.Ek),
                                ss.UMS_fac.subs(subs_vars)))

        self.UB = lru_cache()(lambdify((ss.c, ss.Ek), ss.UB.subs(subs_vars)))

        # van Wyk
        self.UvW = lambdify((ss.c, vw.K, vw.rho_w),
                            vw.U.subs(vw_subs).subs(subs_vars))

        self.vec_UMFint = np.vectorize(self.UMFint)
        self.vec_UMSint = np.vectorize(self.UMSint)
        self.vec_UB = np.vectorize(self.UB)
        self.vec_UvW = np.vectorize(self.UvW)

    def UMFint(self, c):
        return quad(self.UMF_int_inner, 0, self.theta_c(c))[0]

    def UMSint(self, c):
        return dblquad(self.UMS_int,
                       self.theta_c(c), self.theta_ulim,
                       lambda x: self.r_llim, lambda x: self.r_ulim,
                       (c,))[0]

    def prepare(self, c):
        self.UMFint_val = self.vec_UMFint(c)
        self.UMSint_val = self.vec_UMSint(c)

    def integrate(self, c, mu, Ek, K, rho_w):
        UMF = self.UMF_fac_outer(mu, Ek) * (self.UMF_fac_inner(c) *
                                    self.UMFint_val +
                                    self.UMF_second_part(c, mu))
        UMS = self.UMS_fac(Ek) * self.UMSint_val
        UB = self.vec_UB(c, Ek)
```

```
            UvW = self.vec_UvW(c, K, rho_w)
            return UMF, UMS, UB, UvW

        def get_numeric_inner(c, h):
            subs_vars = dict(zip(
                    [kw.d, ss.rd, ss.r0, ss.h, vw.Ef, vw.rho_f, kw.t0],
                    [ss.c, ss.c, data[i][1]-h, h, Ef, rho_f, 0]
                ))

            n = Numeric(subs_vars)
            n.prepare(c)
            return n
        get_numeric = lru_cache()(get_numeric_inner)

        def run_model(c, h, mu, Ek, K, rho_w):
            n = get_numeric(c, h)
            return n.integrate(c, mu, Ek, K, rho_w)

        def f(c, h, mu, Ek, K, rho_w):
            UMF, UMS, UB, UvW = run_model(c*data[i][1], h, mu, Ek, K, rho_w)
            return (2*UMF + UMS + UB + UvW)/((2*data[i][1])**3)

        model = Model(f, independent_vars=['c'])
```

Then we import the data and define initial guesses.

```
In [ ]: def load_data(i):
            c, UT = np.loadtxt('k-compression-csv/%s.csv' % data[i][0],
                               delimiter=',', skiprows=1,
                               usecols = (5, 6), unpack=True)

            c = c[1:]
            UT = UT[1:]

            params = model.make_params()
            params['h'].value = 0.005
            params['h'].min = 0.001
            params['h'].max = data[i][1] * (1-c[-1]) / 2
            params['mu'].value = 0.4
            params['mu'].min = 0
            params['mu'].max = 0.5
            params['Ek'].value = 100 # gf/cm^2
            params['Ek'].min = 0
            params['K'].value = 0.001
            params['K'].min = 0
            params['rho_w'].value = 0.03
            params['rho_w'].min = 0

            return c, UT, params
```

Finally, we perform the fitting itself.

```
In [ ]: results = []
        for i in range(0, len(data)):
            c, UT, params = load_data(i)
            start = time()
            results.append(model.fit(UT, params, c=c))
            print 'Fitting took %d seconds' % (time() - start)

            with open('k-compression-csv/result-fib-%s.txt' %
                      data[i][0], 'w') as text_file:
                text_file.write(results[i].fit_report())
```

And we plot the fit alongside the data (figure 5.44).

```
In [ ]: # TWEAK-START Set to 1 for figure (b)
        i = 0
        # TWEAK-END

        print results[i].fit_report()

        c, UT, _ = load_data(i)

        plt.plot(c, UT,          'b', linewidth=2)
        plt.plot(c, results[i].init_fit, 'k--', linewidth=2)
        plt.plot(c, results[i].best_fit, 'r-', linewidth=1.5)

        ax = plt.gca()
        ax.set_xlabel('Normalised displacement (cm/cm)')
        ax.set_ylabel(r'Energy density (gf.cm/cm$^3$)')
        # TWEAK-START Uncomment to get log plots
        #ax.set_yscale('log')
        # TWEAK-END

        plt.show()
```

We also plot the components of the model, U_S and U_W (figure 5.45):

```
In [ ]: def plot_parts(values):
            h = values['h']
            r0 = data[i][1]-h
            vol = ((2*(r0+h))**3)

            UMF, UMS, UB, UvW = run_model(c*r0, **values)
            UMF = UMF/vol
            UMS = UMS/vol
            UB = UB/vol
            UvW = UvW/vol
            US = 2*UMF + UMS + UB
```

```
        plt.plot(c, UT, 'k--', linewidth=2)
        plt.plot(c, US, 'b', linewidth=2)
        plt.plot(c, UvW, 'r', linewidth=2)

        ax = plt.gca()
        ax.set_xlabel('Normalised displacement (cm/cm)')
        ax.set_ylabel(r'Energy density (gf.cm/cm$^3$)')
        ax.set_yscale('log')

        plt.show()

    plot_parts(results[i].values)
```

B.9 ModifiedKnoppyWebUnitCellFit.ipynb

In this notebook, we implement the modified knoppy web unit cell from section 4.3.2 of the thesis, in a way that can be fitted to experimental data.

We start with imports, as usual.

In [ ]:
```
local = True
```

In [ ]:
```
from ipyparallel import Client
if local:
    rc = Client()
else:
    rc = Client('path/to/ipcontroller-client.json')
dview = rc[:]
dview.clear()

if local:
    # Only run if the engines are on the same machine.
    # On a different machine, make sure to copy equations/
    # and utils.py into the cwd of the engines.
    model_dir = %pwd
    dview.push({'model_dir': model_dir})
    %px import os
    %px os.chdir(model_dir)
```

In [ ]:
```
from __future__ import division
with dview.sync_imports():
    from lmfit import Parameters, minimize
    import numpy as np
    from scipy.integrate import quad, dblquad
    from sympy import lambdify, pprint, symbols
    from time import time

    from equations import simple_sphere as ss, modified_kw as kw
    from equations import vanwyk as vw, factory_params as fp
    from utils import lru_cache
```

```
%px np = numpy
%px ss = simple_sphere
%px kw = modified_kw
%px vw = vanwyk
%px fp = factory_params

from lmfit import Model

import plot as p
import matplotlib.pyplot as plt
```

Constants

We start by defining constants for the model.

Web-specific

The general van Wyk equation has a few constants that we define here.

```
In [ ]: Ef = 3.98 * 10**7 # gf/cm^2 == 3.1 N/tex
        rho_f = 1.3 # g/cm^3
```

Data

We list the data available:

```
In [1]: #    Filename       | rho_k | mwk  | r0 + h | h  | Ek | mu | Scaling factor
        #                   | g/cc  |      | cm     | cm |    |    | (down to ~100
        #                   |       |      |        |    |    |    |   data points)
        data = [
            ('Lot 1 P2',      1/79,  10/90, 0.366, 0.001, 349, 0.5, 6),
            ('Lot 2 Rack2',   1/79,  20/80, 0.366, 0.001, 349, 0.5, 7),
            ('Lot 3 SEM P1',  1/79,  30/70, 0.366, 0.001, 349, 0.5, 5),
            ('Lot 4 #1 P1',   1/79,  20/80, 0.366, 0.001, 349, 0.5, 8),
            ('Lot 5 P1',      1/79,  20/80, 0.366, 0.001, 349, 0.5, 7),
            ('Lot 6 L',       1/79,  20/80, 0.366, 0.001, 349, 0.5, 6),
        ]
```

```
In [ ]: dview.push(dict(Ef=Ef, rho_f=rho_f))
        dview.push(dict(data=data))
```

Helper methods

Next we define some helper methods.

The first helper method calculates c as a function of d and q_c.

```
In [ ]: %%px --local
        def cVal(qc, d, t0):
            cmax = (data[i][3]-(2*data[i][4]))*7/8 # cm
            tmax = t0*7/8
            if d <= cmax and d <= tmax:
                return qc*d
```

```
            elif d <= cmax and d > tmax:
                return qc*d + (1-qc)*(d-tmax)
            elif d > cmax and d <= tmax:
                return qc*cmax
            else:
                return qc*cmax + (1-qc)*(d-tmax)
```

In [ ]:
```
%%px --local
def rdVal(qrd, c, t0):
    return qrd*(c+t0)
```

Fitting the model to data

Now we define the model itself.

In [ ]:
```
%%px --local
vw_outer_subs = dict(zip([vw.v, vw.v0], [kw.vc_outer, kw.v0_outer]))
vw_inner_subs = dict(zip([vw.v, vw.v0], [kw.vc_inner, kw.v0_inner]))

class Numeric(object):
    def __init__(self, subs_vars, cVal, rdVal, t0):
        # c(qc, d, t0)
        self.cVal = lru_cache()(cVal)
        # rd(qrd, c, t0)
        self.rdVal = lru_cache()(rdVal)
        self.t0 = t0

        # Simple sphere

        self.UMF_int_inner = lambdify(ss.theta, ss.UMF_int_inner)
        self.UMS_int = lambdify((ss.r, ss.theta, ss.c, ss.rd),
                                ss.UMS_int.subs(subs_vars))

        self.r_llim = ss.r_llim.subs(subs_vars)
        self.r_ulim = ss.r_ulim.subs(subs_vars)
        self.theta_c = lru_cache()(lambdify((ss.c),
                                            ss.theta_c.subs(subs_vars)))
        self.theta_ulim = ss.theta_ulim.subs(subs_vars)

        self.UMF_fac_inner = lambdify((ss.c, ss.rd),
                                      ss.UMF_fac_inner.subs(subs_vars))
        self.UMF_second_part = lambdify((ss.c, ss.rd),
                                        ss.UMF_second_part.subs(subs_vars))
        self.UMF_fac_outer = lambdify(symbols('y'),
                                      ss.UMF_fac_outer.subs(subs_vars))(0)
        self.UMS_fac = lambdify(symbols('y'), ss.UMS_fac.subs(subs_vars))(0)

        self.UB = lambdify((ss.c), ss.UB.subs(subs_vars))

        # van Wyk
```

```
            self.UvW_outer = lambdify((kw.d, ss.c, vw.K, vw.rho_w),
                                      vw.U.subs(vw_outer_subs).subs(subs_vars))
            self.UvW_inner = lambdify((kw.d, ss.c, ss.rd, vw.K, vw.rho_w),
                                      vw.U.subs(vw_inner_subs).subs(subs_vars))

        def UMFint(self, c):
            return quad(self.UMF_int_inner, 0, self.theta_c(c))[0]

        def UMSint(self, c, rd):
            return dblquad(self.UMS_int,
                           self.theta_c(c), self.theta_ulim,
                           lambda x: self.r_llim, lambda x: self.r_ulim,
                           (c, rd))[0]

        def integrate_UMF(self, d, qc):
            '''Pre-compute the UMF integral so it can be reused'''
            c = self.cVal(qc, d, self.t0)
            self.UMFint_val = self.UMFint(c)

        def integrate_UMS(self, d, qc, qrd):
            '''Pre-compute the UMS integral so it can be reused'''
            c = self.cVal(qc, d, self.t0)
            rd = self.rdVal(qrd, c, self.t0)
            self.UMSint_val = self.UMSint(c, rd)

        def calculate(self, d, qc, qrd, K, rho_w):
            c = self.cVal(qc, d, self.t0)
            rd = self.rdVal(qrd, c, self.t0)

            UMF = self.UMF_fac_outer * (self.UMF_fac_inner(c, rd) *
                                        self.UMFint_val +
                                        self.UMF_second_part(c, rd))
            UMS = self.UMS_fac * self.UMSint_val
            UB = self.UB(c)

            UvW_outer = self.UvW_outer(d, c, K, rho_w)
            UvW_inner = self.UvW_inner(d, c, rd, K, rho_w)

            return UMF, UMS, UB, UvW_outer, UvW_inner


    def calc_rho_w(t0):
        ss_subs_vars = {
            ss.r0: data[i][3] - data[i][4],
            ss.h: data[i][4],
            ss.Ek: data[i][5],
            ss.mu: data[i][6],
            }
```

```
            fp_subs_vars = {
                fp.rho_k: data[i][1],
                fp.t0: t0,
                fp.mwk: data[i][2],
                }

            return lambdify(symbols('y'),
                            fp.rho_w.subs(ss_subs_vars).subs(fp_subs_vars))(0)

        def get_vars(t0):
            ss_subs_vars = {
                ss.r0: data[i][3] - data[i][4],
                ss.h: data[i][4],
                ss.Ek: data[i][5],
                ss.mu: data[i][6],
                }
            vw_subs_vars = {
                vw.Ef: Ef,
                vw.rho_f: rho_f,
                }
            kw_subs_vars = {
                kw.t0: t0
            }

            subs_vars = {}
            subs_vars.update(ss_subs_vars)
            subs_vars.update(vw_subs_vars)
            subs_vars.update(kw_subs_vars)

            return subs_vars

        def evaluate(d, qc, qrd, K, t0):
            rho_w = calc_rho_w(t0)
            subs_vars = get_vars(t0)
            n = Numeric(subs_vars, cVal, rdVal, t0)

            d = d*(data[i][3]+t0)

            n.integrate_UMF(d, qc)
            n.integrate_UMS(d, qc, qrd)
            UMF, UMS, UB, UvW_outer, UvW_inner = n.calculate(d, qc, qrd, K, rho_w)

            vol = ((2*(data[i][3]+t0))**3)
            UMF = UMF/vol
            UMS = UMS/vol
            UB = UB/vol
            UvW_outer = UvW_outer/vol
            UvW_inner = UvW_inner/vol
```

```
            # Calculate U_Tot over the whole
            UTot = 2*UMF + UMS + UB + 2*UvW_outer + UvW_inner

            return UMF, UMS, UB, UvW_outer, UvW_inner, UTot
```

In [ ]:
```
%%px --local
def fmin(pars, d, K, t0):
    parvals = pars.valuesdict()
    qc = parvals['qc']
    qrd = parvals['qrd']
    return evaluate(d, qc, qrd, K, t0)

def calc_min():
    params = Parameters()
    params.add('qc', value=0.5, min=0.001, max=0.999)
    params.add('qrd', value=0.5, min=0.001, max=0.999)

    qc = np.empty(len(d))
    qrd = np.empty(len(d))
    UMF = np.empty(len(d))
    UMS = np.empty(len(d))
    UB = np.empty(len(d))
    UvW_outer = np.empty(len(d))
    UvW_inner = np.empty(len(d))
    UTot = np.empty(len(d))
    #start = time()
    for j in range(0, len(d)):
        minres = minimize(fmin, params, args=(d[j], K, t0), method='lbfgsb')
        qc[j] = minres.params.valuesdict()['qc']
        qrd[j] = minres.params.valuesdict()['qrd']
        UMF[j], UMS[j], UB[j], UvW_outer[j], UvW_inner[j], UTot[j] = fmin(
            minres.params, d[j], K, t0)
    #print 'Minimization took %d seconds' % (time() - start)
    return qc, qrd, UMF, UMS, UB, UvW_outer, UvW_inner, UTot
```

In [ ]:
```
def run_model(d, i, K, t0):
    dview.scatter('d', d)
    dview.push(dict(i=i, K=K, t0=t0))

    start = time()
    %px qc, qrd, UMF, UMS, UB, UvW_outer, UvW_inner, UTot = calc_min()
    print 'Integration took %d seconds' % (time() - start)

    qc = dview.gather('qc').result
    qrd = dview.gather('qrd').result
    UMF = dview.gather('UMF').result
    UMS = dview.gather('UMS').result
    UB = dview.gather('UB').result
    UvW_outer = dview.gather('UvW_outer').result
    UvW_inner = dview.gather('UvW_inner').result
```

```
            UTot = dview.gather('UTot').result

            return qc, qrd, UMF, UMS, UB, UvW_outer, UvW_inner, UTot

        def f(d, i, K, t0):
            _, _, _, _, _, _, _, UTot = run_model(d, i, K, t0)
            return UTot

        model = Model(f, independent_vars=['d', 'i'])
```

Then we import the data and define initial guesses.

```
In [ ]: def load_data(i):
            d, UT = np.loadtxt('compression-csv/%s.csv' % data[i][0],
                               delimiter=',', skiprows=1,
                               usecols = (5, 6), unpack=True)

            d = d[1:]
            UT = UT[1:]

            dfit = d[::data[i][7]]
            UTfit = UT[::data[i][7]]

            return d, UT, dfit, UTfit

        params = model.make_params()
        params['K'].value = 10
        params['K'].min = 0
        params['t0'].value = 0.01 # cm
        params['t0'].min = 0
```

Finally, we perform the fitting itself.

```
In [ ]: results = []
        for i in range(0, len(data)):
            _, _, dfit, UTfit = load_data(i)
            start = time()
            results.append(model.fit(UTfit, params, d=dfit, i=i))
            print 'Fitting for %s took %d seconds' % (data[i][0], time() - start)

            with open('compression-csv/result-%s.txt' % data[i][0], 'w') as text_file:
                text_file.write(results[i].fit_report())
```

And we plot the fit alongside the data (figs. 5.46 to 5.51).

```
In [ ]: # TWEAK-START Data to use
        i = 0
        # TWEAK-END

        print results[i].fit_report()
```

```
d, UT, dfit, UTfit = load_data(i)

plt.plot(dfit, UTfit,         'o', linewidth=2)
plt.plot(dfit, results[i].init_fit, 'k--', linewidth=2)
plt.plot(dfit, results[i].best_fit, 'r-', linewidth=2)

ax = plt.gca()
ax.set_xlabel('Normalised displacement (cm/cm)')
ax.set_ylabel(r'Energy density (gf.cm/cm$^3$)')
# TWEAK-START Uncomment for log plots
#ax.set_yscale('log')
# TWEAK-END
# TWEAK-START Use to adjust figure boundaries
#plt.axis([0, 0.8, 0, 12])
# TWEAK-END

plt.show()
```

Other views

The following views were not used in the thesis, but were useful for checking certain areas of the model.

```
In [ ]: qc, qrd, UMF, UMS, UB, UvW_outer, UvW_inner, UTot = run_model(
            d, i, **results[i].values)
```

```
In [ ]: t0 = results[i].values['t0']
        print data[i][3]/(data[i][3]+t0), t0/(data[i][3]+t0)
```

```
In [ ]: p.plotGraph(d, [UTot, 2*UMF, UMS, UB, UvW],
                    '', 'Normalised displacement (cm/cm)',
                    'Energy density (gf.cm/cm^3)',
                    logNorm=True)
```

```
In [ ]: plt.plot(dfit, UTfit,         'o', linewidth=2)
        plt.plot(d, UTot, 'k--', linewidth=2)
        #plt.plot(d, 2*UMF, linewidth=2)
        #plt.plot(d, UMS, linewidth=2)
        #plt.plot(d, UB, linewidth=2)
        plt.plot(d, 2*UMF+UMS+UB, 'b', linewidth=2)
        #plt.plot(d, 2*UvW_outer, 'b', linewidth=2)
        #plt.plot(d, UvW_inner, 'r', linewidth=2)
        plt.plot(d, 2*UvW_outer+UvW_inner, 'r', linewidth=2)
        #plt.plot(dfit, result.init_fit, 'k--', linewidth=2)
        #plt.plot(dfit, result.best_fit, 'r-', linewidth=1.5)

        ax = plt.gca()
        ax.set_xlabel('Normalised displacement (cm/cm)')
        ax.set_ylabel(r'Energy density (gf.cm/cm$^3$)')
        ax.set_yscale('log')

        plt.show()
```

```
In [ ]: vec_cVal = np.vectorize(cVal)
        vec_rdVal = np.vectorize(rdVal)

        c = vec_cVal(qc, d, t0/(data[i][3]+t0))
        rd = vec_rdVal(qrd, c, t0/(data[i][3]+t0))

        p.plotGraph(d, qc,
                    '', 'Normalised displacement (cm/cm)', r'$q_c$')
        p.plotGraph(d, qrd,
                    '', 'Normalised displacement (cm/cm)', r"$q_{r'}$")
        p.plotGraph(d, c,
                    '', 'Normalised displacement (cm/cm)', r'Normalised $c$ (cm/cm)')
        p.plotGraph(d, rd,
                    '', 'Normalised displacement (cm/cm)', r"Normalised $r'$ (cm/cm)")
```

```
In [ ]: print calc_rho_w(results[i].values['t0'])
```

B.10 vanWykUnitCellFit.ipynb

In this notebook, we fit experimental data for knoppy web compression to the van Wyk model of random fibre assemblies, using the energy method.

We start with imports, as usual.

```
In [ ]: local = True
```

```
In [ ]: from ipyparallel import Client
        if local:
            rc = Client()
        else:
            rc = Client('path/to/ipcontroller-client.json')
        dview = rc[:]
        dview.clear()

        if local:
            # Only run if the engines are on the same machine.
            # On a different machine, make sure to copy equations/
            # and utils.py into the cwd of the engines.
            model_dir = %pwd
            dview.push({'model_dir': model_dir})
            %px import os
            %px os.chdir(model_dir)
```

```
In [ ]: from __future__ import division
        with dview.sync_imports():
            from lmfit import Parameters, minimize
            import numpy as np
            from sympy import lambdify, symbols
            from time import time

            from equations import vanwyk as vw
```

```
        from utils import lru_cache

    %px np = numpy
    %px vw = vanwyk

    from lmfit import Model

    import plot as p
    import matplotlib.pyplot as plt
```

Constants

The general van Wyk equation has a few constants that we define here.

```
In [ ]: Ef = 3.98 * 10**7 # gf/cm^2 == 3.1 N/tex
        rho_f = 1.3 # g/cm^3
```

Data

We list the data available:

```
In [ ]: #    Filename        | Scaling factor
        #                    | (down to ~100 data points)
        data = [
            ('Lot 1 P2',      6),
            ('Lot 2 Rack2',   7),
            ('Lot 3 SEM P1',  5),
            ('Lot 4 #1 P1',   8),
            ('Lot 5 P1',      7),
            ('Lot 6 L',       6),
        ]
```

```
In [ ]: dview.push(dict(Ef=Ef, rho_f=rho_f))
```

Fitting the model to data

Now we define the model itself.

```
In [ ]: %%px --local
        def evaluate():
            # Use unit cell with unit cross-section
            y = symbols('y')
            subs_vars = {
                vw.K:     K,
                vw.Ef:    Ef,
                vw.rho_w: rho_w,
                vw.rho_f: rho_f,
                vw.v0:    1,
                vw.v:     1 - y,
                }
            f = np.vectorize(lambdify(y, vw.U.subs(subs_vars)))
            return f(d)
```

```
In [ ]: def f(d, K, rho_w):
            dview.scatter('d', d)
            dview.push(dict(K=K, rho_w=rho_w))

            %px UvW = evaluate()

            return dview.gather('UvW').result

        model = Model(f, independent_vars=['d'])
```

Then we import the data and define initial guesses.

```
In [ ]: def load_data(i):
            d, UT = np.loadtxt('compression-csv/%s.csv' % data[i][0],
                               delimiter=',', skiprows=1,
                               usecols = (5, 6), unpack=True)

            d = d[1:]
            UT = UT[1:]

            dfit = d[::data[i][1]]
            UTfit = UT[::data[i][1]]

            return d, UT, dfit, UTfit

        params = model.make_params()
        params['K'].value = 40
        params['K'].min = 0
        params['rho_w'].value = 0.001 # g/cm^3
        params['rho_w'].min = 0
```

Finally, we perform the fitting itself.

```
In [ ]: results = []
        for i in range(0, len(data)):
            _, _, dfit, UTfit = load_data(i)
            start = time()
            results.append(model.fit(UTfit, params, d=dfit))
            print 'Fitting for %s took %d seconds' % (data[i][0], time() - start)

            with open('compression-csv/vw-result-%s.txt' %
                      data[i][0], 'w') as text_file:
                text_file.write(results[i].fit_report())
```

And we plot the fit alongside the data (figs. 5.46 to 5.51).

```
In [ ]: # TWEAK-START Data to use
        i = 0
        # TWEAK-END

        print results[i].fit_report()
```

```
d, UT, dfit, UTfit = load_data(i)

plt.plot(dfit, UTfit,         'o', linewidth=2)
plt.plot(dfit, results[i].init_fit, 'k--', linewidth=2)
plt.plot(dfit, results[i].best_fit, 'r-', linewidth=2)

ax = plt.gca()
ax.set_xlabel('Normalised displacement (cm/cm)')
ax.set_ylabel(r'Energy density (gf.cm/cm$^3$)')
# TWEAK-START Uncomment for log plots
#ax.set_yscale('log')
# TWEAK-END
# TWEAK-START Use to adjust figure boundaries
#plt.axis([0, 0.8, 0, 12])
# TWEAK-END

plt.show()
```

B.11 Lee1992.ipynb

The purpose of this notebook is to prove that the implementation of the Lee Carnaby Tandon 1992 model is working correctly. To do this, we recreate various figures from the Lee 1992 paper.

```
In [ ]: titles = False

        from __future__ import division
        import numpy as np
        from scipy.integrate import dblquad
        from sympy import lambdify, pi, symbols

        from equations import lee1992web as lee
        import plot as p
```

Constants

The constants below are taken from Table 1 of [72]. They are used directly in the units provided, which caused some confusion for a while. However the use of grams force combined with grams leads to the answer falling out in N.cm which is sensible enough.

```
In [ ]: Ef = 3.98 * 10**7 # gf/cm^2
        D = 35 * 10**(-4) # cm
        rho = 1.3 # g/cm^3
        Cr = 1.15
        Deff = 0.1 # cm
        m = 0.03 # g/cm^3
        n = 4

        subs_most_vars = dict(zip([lee.Ef, lee.D, lee.rho, lee.Cr, lee.Deff, lee.m],
                                  [Ef, D, rho, Cr, Deff, m]))
```

```
subs_vars = dict(zip([lee.Ef, lee.D, lee.rho, lee.Cr, lee.Deff, lee.m, lee.n],
                     [Ef, D, rho, Cr, Deff, m, n]))

vec_fl = np.vectorize(lambdify((lee.l, lee.n), lee.fl.subs(subs_most_vars)))

gridsize = 120
```

Integration

Now we define the integration itself.

```
In [ ]: class Numeric(object):
            def __init__(self, subs_vars):
                self.N = lambdify((symbols('x')), lee.N.subs(subs_vars))(0)

                self.thetac = lambdify((lee.ea, lee.va), lee.thetac)
                self.lc = lambdify((lee.theta1, lee.ea, lee.va),
                                   lee.lc.subs(subs_vars))

                self.EB_int = lambdify((lee.l1, lee.theta1, lee.ea, lee.va),
                                       lee.EB_int.subs(subs_vars))
                self.ET_int = lambdify((lee.l1, lee.theta1, lee.ea, lee.va),
                                       lee.ET_int.subs(subs_vars))
                self.ES_int = lambdify((lee.l1, lee.theta1, lee.ea, lee.va),
                                       lee.ES_int.subs(subs_vars))

                self.vec_EBint = np.vectorize(self.EBint)
                self.vec_ETint = np.vectorize(self.ETint)
                self.vec_ESint = np.vectorize(self.ESint)

            def EBint(self, ea, va):
                return dblquad(self.EB_int,
                               0, self.thetac(ea, va),
                               lambda x: 0, lambda x: np.inf,
                               (ea, va))[0]

            def ETint(self, ea, va):
                return dblquad(self.ET_int,
                               self.thetac(ea, va), pi/2,
                               lambda x: self.lc(x, ea, va), lambda x: np.inf,
                               (ea, va))[0]

            def ESint(self, ea, va):
                return dblquad(self.ES_int,
                               self.thetac(ea, va), pi/2,
                               lambda x: 0, lambda x: self.lc(x, ea, va),
                               (ea, va))[0]

            def integrate(self, *args):
                EB = self.N * self.vec_EBint(*args)
```

```
        ET = self.N * self.vec_ETint(*args)
        ES = self.N * self.vec_ESint(*args)
        return EB, ET, ES
```

Verification

We start by reproducing Figure 2 (fig. A.3):

```
In [ ]: llim = (0, 0.30) # in cm
        l = np.linspace(*(llim + (gridsize,)))
        fls = []
        for i in [0, 1, 2, 3, 4, 5]:
            fls.append(vec_fl(l, i))
        p.plotGraph(l, fls, titles and 'Figure 4' or None,
                    r'Fiber Segment Length $l$ (cm)', r'$f(l)$ (cm$^{-1}$)',
                    linewidth=1.5,
                    legend=('upper right', [0, 1, 2, 3, 4, 5]));
```

We now move to Figure 4.

```
In [ ]: xlim = (0.05, 0.60)
        va = np.linspace(*(xlim + (gridsize,)))

        ea = -10**(-4)
        EB, ET, ES = Numeric(subs_vars).integrate(ea, va)
```

By scaling the returned energies to μN.cm we can plot Figure 4 (fig. A.4):

```
In [ ]: EB = EB * 10**4
        ET = ET * 10**4
        ES = ES * 10**4
        ETot = EB + ET + ES

        p.plotGraph(va, ETot, titles and 'Figure 4' or None,
                    'Poisson\'s Ratio', r'$E_{Tot}$ (uN.cm)',
                   linewidth=1.5)
```

Moving on to Figure 5, we reproduce it (fig. A.5) in the same manner as Figure 4. In this case, the energies are scaled to mN.cm instead.

```
In [ ]: ea = -10**(-2)
        EB, ET, ES = Numeric(subs_vars).integrate(ea, va)

        EB = EB * 10**1
        ET = ET * 10**1
        ES = ES * 10**1
        ETot = EB + ET + ES

        p.plotGraph(va, ETot, titles and 'Figure 5' or None,
                    'Poisson\'s Ratio', r'$E_{Tot}$ (mN.cm)',
                   linewidth=1.5)
```

Next up: Figure 6. This one requires much more computation, so we will drop down the size of the grid.

```
In [ ]: gridsize = 30
```

First, we create a grid of values between the limits on compressional strain and Poisson's ratio (as taken from the paper). This grid is then used to calculate the energy components (the slow step), which are converted into units of mN.cm.

```
In [ ]: def runGridInt():
            ea = np.linspace(*(eaLim + (gridsize,)))
            va = np.linspace(*(vaLim + (gridsize,)))
            ea, va = np.meshgrid(ea, va)

            EB, ET, ES = Numeric(subs_vars).integrate(ea, va)
            # Scale to mN.cm
            EB = EB * 10**1
            ET = ET * 10**1
            ES = ES * 10**1
            return ea, va, EB, ET, ES

        eaLim = (-0.0001, -0.04)
        vaLim = (0.05, 0.60)
        ea, va, EB, ET, ES = runGridInt()
        ETot = EB + ET + ES
```

We then iterate through the energy grid, extract the minimum energy for each value of compressional strain, and plot the resulting data (fig. A.6).

Note the order of indices used in the search: ν_a is the row index and ϵ_a is the column index. This is an artifact of the np.meshgrid function, which by default returns an array where the y coordinate is the first index, and the x coordinate is the second.

```
In [ ]: def plotMinETot(title, xticks=None):
            eaMin = np.zeros(len(ea))
            minETot = np.zeros(len(ea))
            for i in np.arange(len(minETot)):
                minETot[i] = np.amax(ETot)

            for i in np.arange(len(va)):
                for j in np.arange(len(ea)):
                    if ETot[i,j] < minETot[j]:
                        eaMin[j] = ea[i,j]
                        minETot[j] = ETot[i,j]
            p.plotGraph(-eaMin, minETot, title,
                        'Compressional Strain', r'$min(E_{Tot})$ (mN.cm)',
                       linewidth=1.5,
                       xticks=xticks)

        plotMinETot(titles and 'Figure 6' or None,
                    xticks=[0, 0.01, 0.02, 0.03, 0.04])
```

Analysis

With the data now available, we can go a step further and show full surface and contour plots of the Lee Carnaby model (fig. A.7):

```
In [ ]: p.plotSurface(ea, va, ETot, titles and r'$E_{Tot}$' or None,
                      r'$\epsilon_a$', r'$\nu_a$', r'$E_{Tot}$ (mN.cm)',
                     xticks=[0, -0.01, -0.02, -0.03, -0.04])

        p.plotContour(ea, va, ETot, eaLim, vaLim, titles and r'$E_{Tot}$' or None,
                      r'$\epsilon_a$', r'$\nu_a$', False, True,
                     xticks=[0, -0.01, -0.02, -0.03, -0.04],
                     zlabel=r'$E_{Tot}$ (mN.cm)')
```

Now we extend the above surfaces up to $\epsilon_a = -0.1$ (figs. A.8 and A.9):

```
In [ ]: gridsize = 50
        eaLim = (-0.0001, -0.1)
        ea, va, EB, ET, ES = runGridInt()
        ETot = EB + ET + ES

        plotMinETot(titles and 'Full strain range from Lee Carnaby' or None)
        p.plotSurface(ea, va, ETot, titles and r'$E_{Tot}$' or None,
                      r'$\epsilon_a$', r'$\nu_a$', r'$E_{Tot}$ (mN.cm)')
        p.plotContour(ea, va, ETot, eaLim, vaLim, titles and r'$E_{Tot}$' or None,
                      r'$\epsilon_a$', r'$\nu_a$', False, True,
                     zlabel=r'$E_{Tot}$ (mN.cm)')
```

And then extend the ν_a bounds further (fig. A.10):

```
In [ ]: gridsize = 50
        eaLim = (-0.0001, -0.1)
        vaLim = (0.05, 3)
        ea, va, EB, ET, ES = runGridInt()
        ETot = EB + ET + ES

        plotMinETot(titles and "Expanded range for Poisson's ratio" or None)
        p.plotSurface(ea, va, ETot, titles and r'$E_{Tot}$' or None,
                      r'$\epsilon_a$', r'$\nu_a$', r'$E_{Tot}$ (mN.cm)')
        p.plotContour(ea, va, ETot, eaLim, vaLim, titles and r'$E_{Tot}$' or None,
                      r'$\epsilon_a$', r'$\nu_a$', False, True,
                     zlabel=r'$E_{Tot}$ (mN.cm)')
```

Finally, we examine the overall strain behaviour (fig. A.11):

```
In [ ]: gridsize = 50
        eaLim = (-0.0001, -0.9999)
        vaLim = (0.05, 3)
        ea, va, EB, ET, ES = runGridInt()
        ETot = EB + ET + ES

        p.plotSurface(ea, va, ETot, titles and r'$E_{Tot}$' or None,
```

```
                    r'$\epsilon_a$', r'$\nu_a$', r'$E_{Tot}$ (mN.cm)')
        p.plotContour(ea, va, ETot, eaLim, vaLim, titles and r'$E_{Tot}$' or None,
                    r'$\epsilon_a$', r'$\nu_a$',
                    zlabel=r'$E_{Tot}$ (mN.cm)')
```

Then we plot the corresponding energy components (fig. A.12):

```
In [ ]: p.plotContour(ea, va, EB, eaLim, vaLim, titles and r'$E_B$' or None,
                    r'$\epsilon_a$', r'$\nu_a$',
                    zlabel=r'$E_B$ (mN.cm)')
        p.plotContour(ea, va, ET, eaLim, vaLim, titles and r'$E_T$' or None,
                    r'$\epsilon_a$', r'$\nu_a$',
                    zlabel=r'$E_T$ (mN.cm)')
        p.plotContour(ea, va, ES, eaLim, vaLim, titles and r'$E_S$' or None,
                    r'$\epsilon_a$', r'$\nu_a$',
                    zlabel=r'$E_S$ (mN.cm)')
```

And the corresponding number fractions of different fibre segment states (fig. A.13):

```
In [ ]: class Numeric2(object):
            def __init__(self, subs_vars):
                self.thetac = lambdify((lee.ea, lee.va), lee.thetac)
                self.lc = lambdify((lee.theta1, lee.ea, lee.va),
                                   lee.lc.subs(subs_vars))

                self.NB = lambdify((lee.ea, lee.va), lee.NB.subs(subs_vars))

                self.N_int = lambdify((lee.l1, lee.theta1, lee.ea, lee.va),
                                      lee.N_int.subs(subs_vars))

                self.vec_NB = np.vectorize(self.NB)
                self.vec_NTint = np.vectorize(self.NTint)
                self.vec_NSint = np.vectorize(self.NSint)

            def NTint(self, ea, va):
                return dblquad(self.N_int,
                               self.thetac(ea, va), pi/2,
                               lambda x: self.lc(x, ea, va), lambda x: np.inf,
                               (ea, va))[0]

            def NSint(self, ea, va):
                return dblquad(self.N_int,
                               self.thetac(ea, va), pi/2,
                               lambda x: 0, lambda x: self.lc(x, ea, va),
                               (ea, va))[0]

            def integrate(self, *args):
                NB = self.vec_NB(*args)
                NT = self.vec_NTint(*args)
                NS = self.vec_NSint(*args)
                return NB, NT, NS
```

```
In [ ]: def runGridInt2():
            ea = np.linspace(*(eaLim + (gridsize,)))
            va = np.linspace(*(vaLim + (gridsize,)))
            ea, va = np.meshgrid(ea, va)

            NB, NT, NS = Numeric2(subs_vars).integrate(ea, va)
            return ea, va, NB, NT, NS

        gridsize = 50
        # TWEAK-START Uncomment for (b)
        eaLim = (-0.0001, -0.9999)
        #eaLim = (-0.0001, -0.15)
        # TWEAK-END
        vaLim = (0.05, 3)
        ea, va, NB, NT, NS = runGridInt2()
```

```
In [ ]: p.plotContour(ea, va, NB, eaLim, vaLim, titles and r'$N_B/N$' or None,
                       r'$\epsilon_a$', r'$\nu_a$',
                      zlabel=r'$N_B/N$')
        p.plotContour(ea, va, NT, eaLim, vaLim, titles and r'$N_T/N$' or None,
                       r'$\epsilon_a$', r'$\nu_a$',
                      zlabel=r'$N_T/N$')
        p.plotContour(ea, va, NS, eaLim, vaLim, titles and r'$N_S/N$' or None,
                       r'$\epsilon_a$', r'$\nu_a$',
                      zlabel=r'$N_S/N$')
```

Other views

Here we create a contour plot of θ_c (fig. A.2):

```
In [ ]: gridsize = 60
        eaLim = (0, -1)
        vaLim = (0, 0.60)
        ea = np.linspace(*(eaLim + (gridsize,)))
        va = np.linspace(*(vaLim + (gridsize,)))
        ea, va = np.meshgrid(ea, va)
        thetac = np.vectorize(lambdify((lee.ea, lee.va), lee.thetac))(ea, va)

        p.plotContour(ea, va, thetac, eaLim, vaLim, titles and r'$\theta_c$' or None,
                       r'$\epsilon_a$', r'$\nu_a$',
                      zlabel=r'$\theta_c$')
```

B.12 Supporting files

```
$ equations/
```

```
$ equations/basic_kw.py
```

```
from sympy import *
```

```
from equations import simple_sphere as ss
```

```
d, t0 = symbols('d t_0')


# Conversion between van Wyk and basic knoppy web

b = d - ss.c

t = t0 - b

v0 = (2*(ss.r0+ss.h+t0))**3 - ss.v0

vc = (2*(ss.r0+ss.h+t0))**2 * (2*(ss.rdd+ss.h+t)) - ss.vc
```

```
$ equations/factory_params.py
from sympy import *

from equations import simple_sphere as ss


m_k, N_k, rho_k, mwk, t0 = symbols('m_k N_k rho_k m_wk t_0')

r0 = cbrt( m_k/(8*N_k*rho_k) ) - ss.h

rho_w = mwk * rho_k / ( (1 + t0/(ss.r0 + ss.h))**3 - (pi/6) )
```

```
$ equations/lee1992web.py
# Equations directly from Lee1992

from sympy import *


l1, theta1, l, theta = symbols('l_1 theta_1 l theta')
Ef, D, rho, Cr, Deff, m, n = symbols('Ef D rho C_r Deff m n')
ea, va = symbols('epsilon_a nu_a')


# Part II parameters

L = 4*m / (pi * D**2 * rho) # PII (17)
N = pi * D * L**2 /2        # PII (16)
lbar = L/N                  # PII (19)

c = (1 + ea) / (1 - va*ea) # PII (6)
omega = c**2 / ( sqrt(1 + c**2 * tan(theta)**2) * cos(theta) )**3 # PII (10)

fl = (((n+1)/lbar)**(n+1))/factorial(n) * l**n * exp(-((n+1)/lbar)*l) # PII (13)

k1 = (1 - 1/(Cr**2)) * 2/Deff # PII (15) PI k_1 = PII K_0


# Part I equations

l2 = l1 # PI assumption after (15)

b1 = l1 - l1**3 * k1**2 / 24 # PI (3) using general form

g = sqrt( cos(theta1)**2 * (1+ea)**2 + sin(theta1)**2 * (1-va*ea)**2 ) # PI (28)
```

```
k2 = sqrt( 6*(2/l2)**2 * (1 - (b1*g)/l2) ) # PI (29)

thetac = asin(sqrt( (2 + ea)/((1 + va)*(2 + (1-va)*ea)) )) # PI (30)

lc = (1/k1) * sqrt( 24*(1-(1/g)) ) # PI (31)

B = Ef * pi * D**4 / 64 # PI (52)


# Counts

NB = (1 - cos(thetac)) # PI (41)
N_int = fl.subs(l, l1) * omega.subs(theta, theta1) * sin(theta1)


# Energy terms

EB = (B*l1/2) * (k1-k2)**2 # PI (51)
ET = EB                    # PI (51)
ES = EB.subs(k2, 0)        # PI (53)

# Includes PII modification to ODF
EB_int = EB * fl.subs(l, l1) * omega.subs(theta, theta1) * sin(theta1) # PI (48)
ET_int = ET * fl.subs(l, l1) * omega.subs(theta, theta1) * sin(theta1) # PI (49)
ES_int = ES * fl.subs(l, l1) * omega.subs(theta, theta1) * sin(theta1) # PI (50)

EB_t = N * Integral(EB_int, (l1, 0, oo), (theta1, 0, thetac))     # PI (48)
ET_t = N * Integral(ET_int, (l1, lc, oo), (theta1, thetac, pi/2)) # PI (49)
ES_t = N * Integral(ES_int, (l1, 0, lc), (theta1, thetac, pi/2))  # PI (50)

ETot = EB_t + ET_t + ES_t # PI (45)
```

```
$ equations/modified_kw.py
from sympy import *

from equations import simple_sphere as ss


d, t0 = symbols('d t_0')


# Conversion between van Wyk and modified knoppy web

b = d - ss.c

t = t0 - b

v0_outer = (2*(ss.r0+ss.h+t0))**2 * t0

vc_outer = (2*(ss.r0+ss.h+t0))**2 * t

v0_inner = (2*(ss.r0+ss.h+t0))**2 * 2*(ss.r0+ss.h) - ss.v0

vc_inner = (2*(ss.r0+ss.h+t0))**2 * 2*(ss.rdd+ss.h) - ss.vc
```

```
$ equations/simple_sphere.py
from sympy import *
```

```
r, theta, c, rd, r0 = symbols('r theta c rd r_0')
mu, h, Ek = symbols('mu h E_k')

rdd = r0 - c

theta_c = (pi*c)/(2*r0)


r_llim = r0 - h
r_ulim = r0 + h
theta_ulim = pi/2


v0 = 4 * pi * (r0+h)**3 / 3
vc = pi/3 * (rdd+h) * (6*rd**2 + 3*pi*rd*(rdd+h) + 4*(rdd+h)**2)


# UMF

erho = rd/(theta_c*r0) - 1
ephi = (theta/sin(theta))*(1+erho) - 1

erho_term = erho**2 * r
ephi_term = ephi**2 * r
cross_term = 2*mu*erho*ephi*r

UMF_int = (erho**2 + ephi**2 + 2*mu*erho*ephi) * r**2 * sin(theta)
UMF_fac = pi * Ek / (1 - mu**2)

UMF_int_inner = theta**2 / sin(theta)
UMF_fac_inner = (1+erho)**2
UMF_second_part = 2*(1 + erho**2 - 2*mu*erho)*(sin(theta_c/2)**2) + \
                  2*(1+erho)*(mu*erho-1)*(theta_c**2)/2
UMF_fac_outer = 2 * pi * Ek * h * (h**2 + 3*r0**2) / (3*(1 - mu**2))


# UMS

theta_f_old = (pi/2 - theta) * (r/(r-c))
theta_f = (pi/2 - theta) * (r0/(r0-c))
rho_o_i = r*sin(theta)
z_i = r*cos(theta)
rho_o_f_old = rd + (r-c) * cos(theta_f_old)
rho_o_f = rd + (r-c) * cos(theta_f)
z_f = (r-c) * sin(theta_f)

UMS_ephi_old = (rho_o_f_old / rho_o_i) - 1
UMS_ephi = (rho_o_f / rho_o_i) - 1

UMS_int = UMS_ephi**2 * r**2 * sin(theta)

UMS_fac = 2*pi*Ek


# UB

k1 = 1/r0
k2 = 1/(r0-c)

UB = 4 * pi * Ek * (h**3 * r0**2 / 3 + h**5 / 5) * (k1-k2)**2 * cos(theta_c)
```

$ equations/vanwyk.py

```
from sympy import *


K, Ef, rho_w, rho_f = symbols('K E_f rho_w rho_f')
v, v0 = symbols('v, v_0')


# Integral of dU = -p dv

factor = K * Ef * (rho_w / rho_f)**3

U = factor * ((v0**3)/(2*v**2) + v - 3*v0/2)
```

$ plot.py

```
from __future__ import division
from matplotlib import cm, gridspec, patches, rc
from matplotlib.colors import LogNorm
from matplotlib.image import NonUniformImage
from matplotlib.mlab import griddata
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

rc('font', size=15)


def plotGraph(x, y, title, xlabel, ylabel, logY=False, logX=False, linewidth=1,
              legend=None, xticks=None):
    fig = plt.figure()
    ax = fig.add_subplot(111)
    if type(y) == type([]):
        for i in range(0, len(y)):
            if legend:
                ax.plot(x, y[i], linewidth=linewidth, label=legend[1][i])
            else:
                ax.plot(x, y[i], linewidth=linewidth)
    else:
        ax.plot(x, y, linewidth=linewidth)

    if title:
        ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)

    if logY:
        ax.set_yscale('log')
    if logX:
        ax.set_xscale('log')
    if legend:
        ax.legend(loc=legend[0])
    if xticks:
        ax.set_xticks(xticks)

    plt.show()

def plotSurface(x, y, z, title, xlabel, ylabel, zlabel, logNorm=False,
                zlim=None, xticks=None):
    norm = None if not logNorm else LogNorm()
```

```
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    if zlim:
        for i in np.arange(z.shape[0]):
            for j in np.arange(z.shape[1]):
                if z[i, j] < zlim[0] or z[i, j] > zlim[1]:
                    z[i, j] = np.NaN

    surf = ax.plot_surface(x, y, z,
                           rstride=1,
                           cstride=1,
                           cmap=cm.coolwarm,
                           norm=norm,
                           linewidth=0,
                           antialiased=False)

    if title:
        ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    ax.set_zlabel(zlabel)
    # TODO fix when 1.4.4 is released
    ax.xaxis._axinfo['label']['space_factor'] = 1.8
    ax.yaxis._axinfo['label']['space_factor'] = 1.8
    ax.zaxis._axinfo['label']['space_factor'] = 2
    if zlim:
        ax.set_zlim(*zlim)
    if xticks:
        ax.set_xticks(xticks)

    # Remove outer whitespace
    plt.tight_layout()
    plt.show()

def plotContour(x, y, z, xlim, ylim, title, xlabel, ylabel, logNorm=False,
                showMin=False, reshape=False, extraLine=None, xticks=None,
                zlabel=None, reshapeInterp='linear'):
    """Plot the provided data as a contour plot.

    The data must use the xy indexing.
    """
    norm = None if not logNorm else LogNorm()

    fig = plt.figure()
    ax = fig.add_subplot(111)
    cset = ax.contour(x, y, z,
                      zdir='z',
                      linewidths=2,
                      cmap=cm.gnuplot,
                      norm=norm)
    if zlabel:
        fig.colorbar(cset, label=zlabel)
    else:
        fig.colorbar(cset)

    if (reshape):
        xi = np.reshape(x, np.prod(x.shape))
        yi = np.reshape(y, np.prod(y.shape))
        zi = np.reshape(z, np.prod(z.shape))
        xf = np.linspace(*(xlim + (x.shape[1],)))
```

```
        yf = np.linspace(*(xlim + (y.shape[0],)))
        zf = griddata(xi, yi, zi, xf, yf, interp=reshapeInterp)
        im = NonUniformImage(ax,
                    interpolation='bilinear',
                    origin='lower',
                    extent=xlim + ylim,
                    cmap=cm.coolwarm,
                    norm=norm)
        im.set_data(xf, yf, zf)
        ax.images.append(im)
    else:
        ax.imshow(z,
                    interpolation='bilinear',
                    origin='lower',
                    extent=xlim + ylim,
                    aspect='auto',
                    cmap=cm.coolwarm,
                    norm=norm)

    if title:
        ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)

    if showMin:
        xmin = np.zeros(x.shape[1])
        ymin = np.zeros(x.shape[1])
        zmin = np.empty(x.shape[1])
        zmin[:] = np.amax(z)

        for i in np.arange(y.shape[0]):
            for j in np.arange(len(xmin)):
                if z[i,j] < zmin[j]:
                    xmin[j] = x[i,j]
                    ymin[j] = y[i,j]
                    zmin[j] = z[i,j]

        ax.plot(xmin, ymin, c='black', ls='--', lw=2.0)

    if extraLine:
        if type(extraLine) == type([]):
            for i in extraLine:
                ax.plot(i[0], i[1], c='blue', ls='-.', lw=3.0)
        else:
            ax.plot(extraLine[0], extraLine[1], c='blue', ls='-.', lw=3.0)

    if xticks:
        ax.set_xticks(xticks)

    plt.show()

def draw_sphere(axis, x0, y0, rd, rdd, h, full=True):
    outer_width = 2*(rdd + h)
    inner_width = outer_width - 4*h

    if (full):
        left_inner = patches.Arc((x0-rd, y0), inner_width, inner_width,
                                 180, 270, 90)
        left_outer = patches.Arc((x0-rd, y0), outer_width, outer_width,
                                 180, 270, 90)
        axis.add_patch(left_inner)
```

```
        axis.add_patch(left_outer)

    right_inner = patches.Arc((x0+rd, y0), inner_width, inner_width,
                              180, 90, 270)
    right_outer = patches.Arc((x0+rd, y0), outer_width, outer_width,
                              180, 90, 270)
    axis.add_patch(right_inner)
    axis.add_patch(right_outer)

    if (rd > 0):
        left_edge = full and -rd or 0
        top_inner = plt.Line2D((x0+left_edge, x0+rd), (y0+rdd-h, y0+rdd-h),
                               c='black')
        top_outer = plt.Line2D((x0+left_edge, x0+rd), (y0+rdd+h, y0+rdd+h),
                               c='black')
        axis.add_line(top_inner)
        axis.add_line(top_outer)

        bottom_inner = plt.Line2D((x0+left_edge, x0+rd), (y0-rdd+h, y0-rdd+h),
                                  c='black')
        bottom_outer = plt.Line2D((x0+left_edge, x0+rd), (y0-rdd-h, y0-rdd-h),
                                  c='black')
        axis.add_line(bottom_inner)
        axis.add_line(bottom_outer)

def draw_vanwyk_web(axis, x0, y0, web_width, web_height):
    x_offset = web_width / 2
    y_offset = web_height / 2

    rect = patches.Rectangle((x0-x_offset, y0-y_offset),
                             web_width, web_height, fill=False)
    axis.add_patch(rect)

def draw_web(axis, rdd, t, t0, va, web_width):
    x_offset = web_width / 2
    scaling_factor = 1 - ((t-t0)/t0)*va

    top = patches.Rectangle((-x_offset*scaling_factor, rdd),
                            web_width*scaling_factor, t, fill=False)
    bottom = patches.Rectangle((-x_offset*scaling_factor, -rdd-t),
                            web_width*scaling_factor, t, fill=False)
    axis.add_patch(top)
    axis.add_patch(bottom)

def plot_simple_sphere(rd, rdd, h, plot_range, energies):
    if (energies):
        gs = gridspec.GridSpec(1, 2, width_ratios=[3, 1])
        plt.subplot(gs[0])
    else:
        plt.axes()

    draw_sphere(plt.gca(), 0, 0, rd, rdd, h)
    plt.axis(plot_range)
    plt.gca().set_aspect(1)

    if (energies):
        plt.subplot(gs[1])
        plt.gca().bar(np.arange(len(energies)), energies, 0.5)

    plt.show()
```

```
def plot_basic_kw_unitcell(rd, rdd, h, web_width, web_height, plot_range,
                           energies):
    if (energies):
        gs = gridspec.GridSpec(1, 2, width_ratios=[3, 1])
        plt.subplot(gs[0])
    else:
        plt.axes()

    draw_sphere(plt.gca(), 0, 0, rd, rdd, h)
    draw_vanwyk_web(plt.gca(), 0, 0, web_width, web_height)

    plt.axis(plot_range)
    plt.gca().set_aspect(1)

    if (energies):
        plt.subplot(gs[1])
        plt.gca().bar(np.arange(len(energies)), energies, 0.5)

    plt.show()

def plot_simple_kw_unitcell(rd, rdd, h, t, t0, va, web_width, plot_range):
    plt.axes()

    draw_sphere(plt.gca(), 0, 0, rd, rdd, h)
    draw_web(plt.gca(), rdd, t, t0, va, web_width)

    plt.axis(plot_range)
    plt.gca().set_aspect(1)
    plt.show()

def plot_basic_kw(kw_width, kw_height, rd, rdd, h, web_width, web_height,
                  plot_range, energies):
    if (energies):
        gs = gridspec.GridSpec(1, 2, width_ratios=[3, 1])
        plt.subplot(gs[0])
    else:
        plt.axes()

    for i in range(0, kw_width):
        for j in range(0, kw_height):
            x0 = i*web_width - web_width*(kw_width-1)/2
            y0 = j*web_height - web_height*(kw_height-1)/2
            draw_sphere(plt.gca(), x0, y0, rd, rdd, h)
            draw_vanwyk_web(plt.gca(), x0, y0, web_width, web_height)

    plt.axis(plot_range)
    plt.gca().set_aspect(1)

    if (energies):
        plt.subplot(gs[1])
        plt.gca().bar(np.arange(len(energies)), energies, 0.5)

    plt.show()
```

$ utils.py

```
import time
import functools
import collections

def lru_cache(maxsize = 255, timeout = None):
```

```
    """lru_cache(maxsize = 255, timeout = None) --> returns a decorator which
    returns an instance (a descriptor).

    Purpose   - This decorator factory will wrap a function / instance method and
                will supply a caching mechanism to the function. For every given
                input params it will store the result in a queue of maxsize size,
                and will return a cached ret_val if the same parameters are
                passed.

    Params    - maxsize - int, the cache size limit, anything added above that
                          will delete the first values enterred (FIFO). This
                          size is per instance, thus 1000 instances with maxsize
                          of 255, will contain at max 255K elements.
              - timeout - int / float / None, every n seconds the cache is
                          deleted, regardless of usage. If None - cache will
                          never be refreshed.

    Notes     - If an instance method is wrapped, each instance will have it's
                own cache and it's own timeout.
              - The wrapped function will have a cache_clear variable inserted
                into it and may be called to clear it's specific cache.
              - The wrapped function will maintain the original function's
                docstring and name (wraps)
              - The type of the wrapped function will no longer be that of a
                function but either an instance of _LRU_Cache_class or a
                functool.partial type.

    On Error - No error handling is done, in case an exception is raised - it
                will permeate up.
    """

    class _LRU_Cache_class(object):
        def __init__(self, input_func, max_size, timeout):
            self._input_func        = input_func
            self._max_size          = max_size
            self._timeout           = timeout

            # This will store the cache for this function, format -
            # {caller1 : [OrderedDict1, last_refresh_time1],
            #  caller2 : [OrderedDict2, last_refresh_time2]}.
            # In case of an instance method - the caller is the instance, in
            # case called from a regular function - the caller is None.
            self._caches_dict       = {}

        def cache_clear(self, caller = None):
            # Remove the cache for the caller, only if exists:
            if caller in self._caches_dict:
                del self._caches_dict[caller]
                self._caches_dict[caller] = [collections.OrderedDict(), time.time()]

        def __get__(self, obj, objtype):
            """ Called for instance methods """
            return_func = functools.partial(self._cache_wrapper, obj)
            return_func.cache_clear = functools.partial(self.cache_clear, obj)
            # Return the wrapped function and wraps it to maintain the docstring
            # and the name of the original function:
            return functools.wraps(self._input_func)(return_func)

        def __call__(self, *args, **kwargs):
            """ Called for regular functions """
            return self._cache_wrapper(None, *args, **kwargs)
```

```
        # Set the cache_clear function in the __call__ operator:
        __call__.cache_clear = cache_clear


        def _cache_wrapper(self, caller, *args, **kwargs):
            # Create a unique key including the types (in order to differentiate
            # between 1 and '1'):
            kwargs_key = "".join(map(
                    lambda x : str(x) + str(type(kwargs[x])) + str(kwargs[x]),
                    sorted(kwargs)))
            key = "".join(map(lambda x : str(type(x)) + str(x) , args)) + kwargs_key

            # Check if caller exists, if not create one:
            if caller not in self._caches_dict:
                self._caches_dict[caller] = [collections.OrderedDict(), time.time()]
            else:
                # Validate in case the refresh time has passed:
                if self._timeout != None:
                    if time.time() - self._caches_dict[caller][1] > self._timeout:
                        self.cache_clear(caller)

            # Check if the key exists, if so - return it:
            cur_caller_cache_dict = self._caches_dict[caller][0]
            if key in cur_caller_cache_dict:
                return cur_caller_cache_dict[key]

            # Validate we didn't exceed the max_size:
            if len(cur_caller_cache_dict) >= self._max_size:
                # Delete the first item in the dict:
                cur_caller_cache_dict.popitem(False)

            # Call the function and store the data in the cache (call it with
            # the caller in case it's an instance function - Ternary condition):
            cur_caller_cache_dict[key] = self._input_func(caller, *args, **kwargs) \
                    if caller != None else self._input_func(*args, **kwargs)
            return cur_caller_cache_dict[key]


    # Return the decorator wrapping the class (also wraps the instance to
    # maintain the docstring and the name of the original function):
    return (lambda input_func : functools.wraps(input_func)(
            _LRU_Cache_class(input_func, maxsize, timeout)))
```